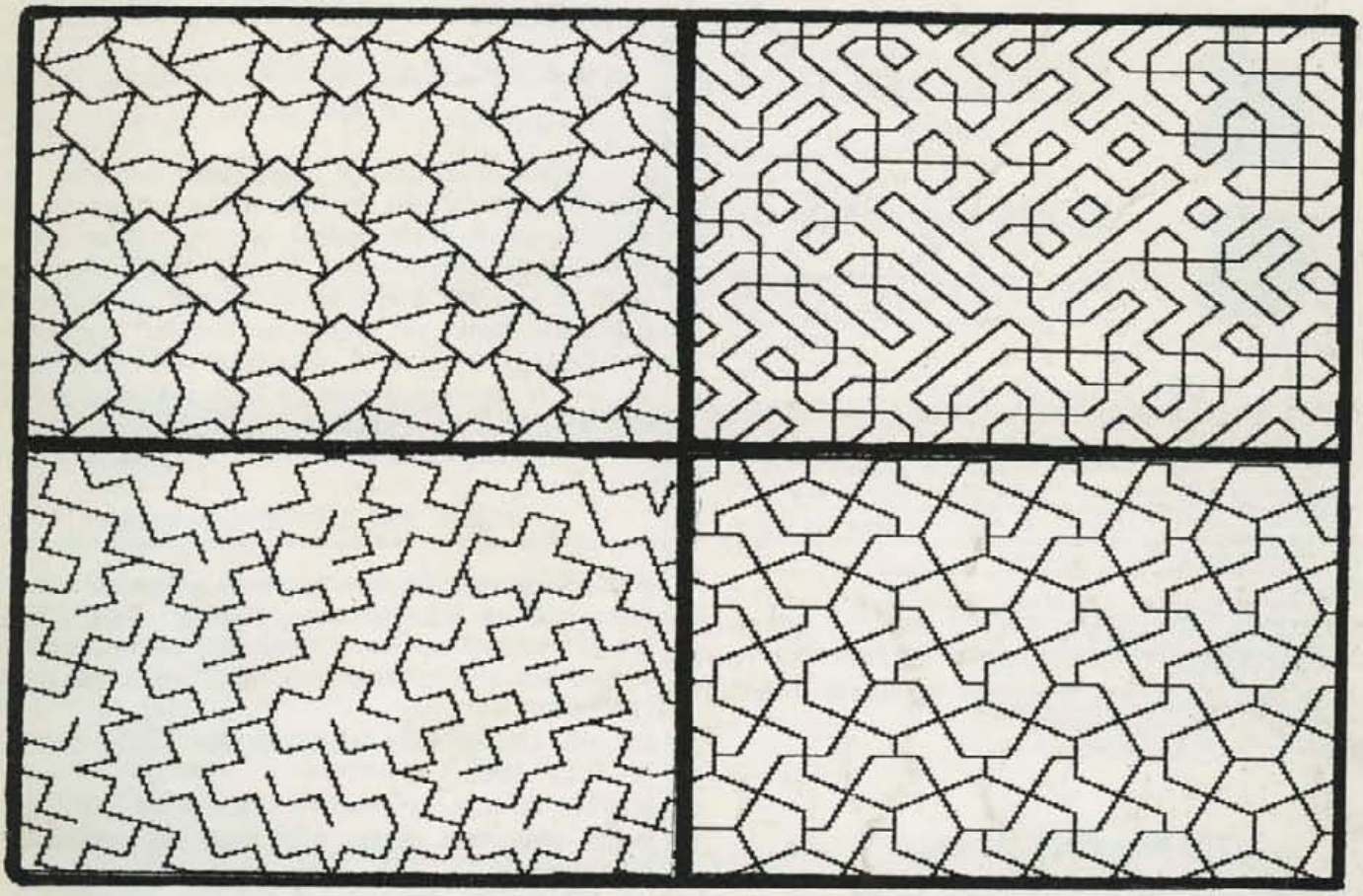
hard core

THE JOURNAL OF THE BRITISH APPLE SYSTEMS USER GROUP

JUNE 1982 £1 VOLUME 2·No. 3



hard core

THE JOURNAL OF
THE BRITISH APPLE SYSTEMS USER GROUP
P.O.BOX 174 WATFORD WD2 6NF

EDITED BY DAVID BOLTON

CONTENTS

Cover Illustration - Some examples of the output from R.A.Fairthorne's RANDOM TILING program, featured in this issue.

EDITORIAL

Looking back over the eight issues so far of HARDCORE, there has been a consistent trend of improvement in the size and quality of the magazine in general. There has, unfortunately, also been a consistent trend of deterioration in the size of the editorials, as deadlines loom with persistent, if not monotonous, regularity. Both these trends, I am pleased in the former case and ashamed in the latter case to say, continue!

David Bolton

GAME REVIEW
****

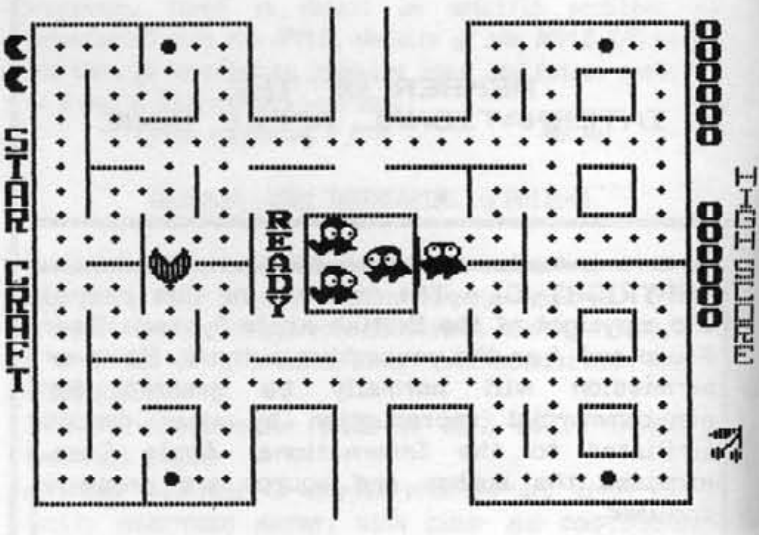
SNOGGLE
****

BY STUART MORLEY

Snoggle is a fast hires arcade game for 1 player, when the disk is booted on either a 13 or 16 sector Apple the program starts by issuing an attractive title page, as this page is displayed the main program loads, to the sound of strange clicking noises. After about 15-20 seconds the main title page is displayed and the game commences. The object of the game is simple , the player (denoted by a round object with a mouth that opens and closes) has to travel around a small maze as shown in the picture. The puckman (player) is steered by four keys :- A,Z,<-,-> Up,down,left,right respectively. The puckman must eat all of the dots in the maze to be allowed to proceed to the next level. However to make the game more difficult four ghosts also inhabit the maze, which may sound alright, but they have a tendency to eat you if you are not careful and bump into them !!! You do however have at your disposal four dots that are placed in all four corners of the screen. These help in the following ways : If you travel over these dots then the ghosts become almost invisable for about 5 seconds in which time, if you manage to bump into them, you eat them (it makes a change for you to eat the ghosts - not be eaten !!!). If you manage to eat the ghosts then you score points - 200pts for the first 400pts for the second 800pts for the third and 1600pts !! for the fourth. However if you do manage to get the better of them they will still be reborn again in the centre of the maze. The game consists of seven levels each represented by a fruit which also appears near the centre of the maze at random times allowing the puckman to eat it and receive more points. The first level is fairly difficult and after having the game for more than two months I have still not managed to get further than the 'SECOND' level with only 10800pts!!!. The documentation does say that at first the direction keys aren't very responsive, this however as explained in the manual is due to your timing, you will after 10 or so games be able to handle the puckman much better.

I have found that out of all the games I have I always seem to come back to Snoggle & never seem to either beat it or get bored with it.

If you are frustrated by programs that are not easily mastered then don't buy Snoggle !!!



LOCAL GROUPS

LEICESTER APPLE USERS GROUP FOR HELP AND SUPPORT (L.A.U.G.H.S) (Affiliated to BASUG)

This is a thriving group of lively Apple Users which started nine months ago. With our increasing membership we have now outgrown our present premises at the 'Leicester Computer Centre' and from 7th April 1982 we have been meeting at:

The Winstanley Arms
The Glade
Narborough Road
Leicester
(Nr M1 Junction 21)

Our meetings vary in content and include interesting discussions and speaklers, demonstrations of software and "Help Forums". Members' facilities include (embryonic) software and text libraries and discounts on computer products.
If you are a BASUG member or an Apple User living in the East Midlands area you would be welcome at our meetings which start at 7.30 p.m. on the 1st Wednesday of the month.
If you require further information this can be obtained from the Secretary:

Hazel Brown
7 Bude Drive
Glenfield
Leicester
Tel. 0533 875253

or
Mike Preston (Chairman) Tel. 7774 647

CROYDON APPLE USER GROUP

May we through you kind offices provide further details of our Group? We meet at 7.00 p.m. on the 2nd Monday of the month.
We are all professionally involved in Apples or ITT 2020's, nearly all do some programming, but are not impartial to producing and discussing games programs. Recent illustrated lectures have been on Structured Programming, Databases and Apple Comms.
As our venue is still unsettled, readers might care to phone me at 01-777 5478 evenings for details.

Paul Vernon
Hon Sec.

BASUG SOUTH LONDON GROUP

Report by Tony Williams

Eagle-eyed basuggers will at once have spotted the subtle renaming of the Raynes Park Group. Members recently did a quick tally of themselves and concluded that lo! they did not come from South West London at all - but from all over, actually. No shillyshalliers they, the modest title of "South London" was assumed forthwith. Meetings will continue to be held in the

Raynes Park Methodist church.
Worple Rd
Entrance 2nd door up Tolverne Rd

News in brief. Warren Avery has bowed out as chairman after a year or so's valiant and selfless service. Into his shoes stepped Michael Leeming. Tony Freedman is Hon Treas. and Neil Stephenson Secretary. I think. An organized bunch this lot. They already know their programme through to December! And here are the dates you need to know:

8th July: Printers
9th Sept: Microcomputers in Education (Jim Brannin)
14th Oct: Communications and Networking.
11th November: What do I do with my slots? (i.e. peripherals and that)
9th Dec: Games.

South London has stabilized its meetings to the second Thursday in every month, except August. Neil Stephenson has promised to write a round up of reports on meetings already held, including Ian Trackman's "Ins and Outs of Machine Code" presentation on May 13.

B.A.S.U.G

SOUTH LONDON GROUP

Raynes Park Methodist Church
Worple Road
Entrance 2nd Door Up Tolverne Rd.

2nd Thursday Of Each Month at 7-30 PM

10 June BASUG Software Library.

8 July Printers.

9 Sept Micros in Education.-Jim Brannin.

14 Oct Communications and Networking.

11 Nov What Do I Do With My Slots.

9 Dec Games.

BEGINNERS PAGE

by John Sharp

At the end of the last Beginners Page, I posed the problem of printing the letters of the alphabet in pairs, AZ BY etc. Before going that far, how are we going to print the alphabet at all using a loop. We want to say:

```
PRINT "ABCDE.....WXYZ"
```

and the APPLE does not allow loops like

```
10 FOR N$ = "A" TO "Z"
20 PRINT N$
30 NEXT N$
```

FOR=NEXT loops only work with numbers, as the computer can only work by counting, so we have to find a numerical code for the letters.

APPLESOFT (but not INTEGER BASIC) has a command to convert numbers to letters :- CHR$(N) where N is a number or numerical variable. IF N= 65 then CHR$(N) = "A" and so on up to 90 when CHR$(N) = "Z".
So, to print all the letters we have the simple program

```
10 FOR N = 65 TO 90
20 PRINT CHR$(N)
30 NEXT N
```

If we make line 20 PRINT CHR$(N); they will print next to one another on the same line.
Now when we want to print not ABCDEF.... but AZ BY CX etc., we have two movements of the numbers so to speak. We have to find a way to allow the number to increase to give us the A to Z part and at the same time decrease so that we can run backwards. The decrease is regular; if we look in terms of the numbers we see :-

```
AZ    CHR$(65);CHR$(90)
BY    CHR$(66);CHR$(89)
:     :        :
:     :        :
```

The numbers are increasing by one in the first column and decreasing by one in the other. The numbers are not related to one another or are they? If you look closely you will see that 65 + 90 = 155 as does 66 + 89 and so on. So the general case would be CHR$(N) ; CHR$(155-N).
Thus it is really quite easy. We replace line 20 in the above program by

```
20 PRINT CHR$(65 + N); CHR$(155) - N
```

Thanks to Vernon Quaintance for that solution. There is another way to use a similar loop:-

```
10 FOR N = 0 to 25
20 PRINT CHR$(65 + N); CHR$(90 -N)
30 NEXT
```

Yet another way would be :-

```
10 LET M = 92
20 FOR N = 65 TO 90
30 M = M -1
40 PRINT CHR$(N); CHR$(M)
50 NEXT N
```

This isn't too elegant in this particular case; the other two are better. However, if the two are not connected we might need to use this sort of approach e.g.

```
10 FOR N = 1 to 10
20 M = M + 2
30 PRINT CHR$(N); CHR$(M)
40 NEXT N
```

So loops require juggling with numbers. A training in mathematical puzzles gives one a distinct advantage in such programming. By seeing such connections, very succinct programs can be obtained.

Let's now build up a set of loops to show what I mean.
Suppose we wish to "draw" a checkerboard using the screen in the text mode. It would be similar on the GRAPHICS screen.
We will use a character "*". As the screen is 24 characters high, we can use blocks of 3. Starting with three '*'s means a loop thus:

```
200 FOR N = 1 to 3
210 PRINT "*";
220 NEXT N
```

Now if we build three rows of these by adding:-

```
150 FOR M = 1 to 3
250 NEXT M
```

Remember that the loop for M must be outside that for the N loop.

But wait a minute , this prints all in one line, since line 210 ends in a ";".
We need to send a "go back to the next line" command before going on. That is outside the "N" loop but inside the "M" loop. We can do this by adding:-

```
230 PRINT
```

Now we wish to move these blocks about the screen leaving a gap of three spaces. Adding the lines:-

```
100 FOR L = 0 TO 18 STEP 6
300 NEXT L
```

The step 6 makes sure that 3 spaces are added as well.
To make the movement across, we can make use of the HTAB command instead. BUT we must make sure that we travel the distance moved by the N loop as well as that of the L loop, otherwise we print on top of one another. If we add the movement down with a VTAB, we can remove the PRINT command in line 230.
Line 10 has been added to see the picture cleanly, and slow it down to see exactly what is going on as the computer is moving too fast for the eye.

```
10 HOME : SPEED = 100
100 FOR L = 0 TO 18 STEP 6
150 FOR M = 1 TO 3
200 FOR N = 1 TO 3
205 HTAB L + N
207 VTAB M
210 PRINT "*";
220 NEXT N
230 NEXT M
250 NEXT L
```

Now we need to produce the rows. If we add

```
80 FOR K = 0 TO 21 STEP 3
400 NEXT K
```

and alter line 207 to VTAB M + K

This gives a downward movement, but not the familiar checkerboard. We need to check if we are on an odd or an even row, to see whether we start at the end or at the next block along. So adding a some new lines 85/90 and altering 100 to take account of the changes gives the final program as follows.

```
10 HOME: SPEED = 100
80 FOR K = 0 TO 21 STEP 3
85 IF ( (K/2) - (INT k/2 + .005) > 0.4
   THEN A = 0: B = 18: GOTO 100
90 A = 3:B = 21
100 FOR L = A TO B STEP 6
150 FOR M = 1 TO 3
200 FOR N = 1 TO 3
205 HTAB N +L
207 VTAB K + M
210 PRINT "*";
250 NEXT M
300 NEXT L
400 NEXT K
450 IF PEEK (-16384) < 127 THEN 450
```

Thus it is quite a short program, if tackled in the right way, with the computer doing the work. I was once asked to show someone how to do this, (not an APPLE) because despite having a 4K machine, he had run out of memory!

What about that last line? Well leave it out and see what happens.

EPSON PRINTER PAGES

by Quentin Reidford

/To ease the burden on John Sharp and the other stalwarts in the deep south Quentin of Sheffield has volunteered to take up writing the Epson Pages./

As John Sharp mentioned in his article in the February '82 'Hardcore' the main source of erratic printing problems lies in the use of different PROMS on the interface card. My card uses the 'D' version of the PROM and has worked 'by the book' - and this includes the EMPHASISED driver submitted by John. Some curiosities, however, include the use of TABS in print statements. I find that I have to use the TAB (n) mode - counting the value of (n) from the last printed position on that line, plus the length of any string on that line. An example of this can be found in the useful utility by John Sharp in the Feb. issue - the Catalog Label Printer. This requires a variable (T) to allow tabbing.

```
685 T = LEN (NAME$(Z)): REM ***
Tabbing constant on first field

690 PRINT NAME$(Z); TAB( 80 - T)
;NAME$(Z + INT (N/2) + P1
```

Perhaps the main disappointment in the MX-80, however, is in the distortion of graphics. The printer, in its F/T 2 guise, will, as you know, dump either of the Hi-res screens through software. On the F/T 80, however, the gearing of the print head is such that it travels too quickly resulting, for instance, in ellipses rather than circles being printed. Even more irritating is the inability to print the whole screen in the EXTENDED mode. If anyone has any ideas that they wish to share, PLEASE write in.

Perhaps the most newsworthy item is the availability of the LCC 8132 EPSON EPROM from Leicester Computer Centre.

This EPROM is a direct replacement for the PROM in location 4a on the Epson Apple interface card. I have not had time to make a thorough review of its capabilities in this issue, but briefly it will allow the Epson to behave with programs like VISICALC, DB-MASTER and, apparently, APPLE-SPELLER.

This is because the Prom obeys normal printer conventions e.g. - Ctrl I 80 N and can therefore be recognised by the software. Good news for those who have the misfortune of having DB MASTER and an Epson. At approx. £18.00 + VAT this EPROM is cheaper than going out for an Centronics.

Other features include the use of CTRL characters instead of POKES for graphics dumps.

CTRL I G 2 E I <RETURN>, for instance, will dump the contents of <G>raphics, page <2>, <E>xpanded and <I>nversed - easier to remember too.

The PROM comes with a slim plastic-bound booklet explaining the options and changes and complete with listings for three demonstration programs.
I have had this one in for about four hours and I am now forced to think of plausible excuses to give my wife that it is an essential acquisition....Seriously though, I would think that it has to be a must, especially for those with the Revision 'C' and earlier Epson PROM.

The other EPSON utility program which is on the market is OMNIFONT. This allows the use of all the fonts from the DOS TOOLKIT within a printing program, with up to four of those fonts within the same program. The disc contains a modified driver for Apple Writer allowing the user to print a letter in any font, either your own or one from the toolkit. Obviously to make use of the whole package you require both Apple Writer and the Toolkit. The software supports EXTENDED, CONDENSED, UNDERLINED and INVERSE printing, although you can get some odd results trying to justify in the extended mode. The BASIC program's ANIMATRIX module enables the user to print out logos or symbols, although you need to use the VARIABLE LINE SPACING algorithm to ensure that all the pieces join up. Any of you who have experienced the brain-ache in converting Binary to Hex. for Bit-image printing will appreciate this feature...

Initial versions of the program had either a bug or possibly only an unusual logic switch which caused the Applewriter section to crash. Happily, later versions have cured this and the program works beautifully, although using the standard program i.e., without Apple Writer option can be hard going!

Finally, for those who were lucky enough NOT to experience any difficulty in using the ENHANCED driver. If you have been using it to print from a constant address of 300 at option 'J' then you will find that if you change 'J' back to C100 your file will be printed out in CONDENSED mode. However it will only work AFTER you have PRINTED in Enhanced mode first. And don't forget to alter your right margin !....

Please write to me c/o BASUG and I will try to answer any problems. PLEASE send any solutions to problems that you have had and managed to overcome....If you DO have a problem then be sure to include the DOS version you use and if possible the PROM version in your card.

My thanks to MICROSOURCE for allowing me review copies of both OMNIFONT and the LCC 8132 EPSON EPROM.

There was a mistake in the listing of CATALABEL program by John Sharp in February's HARDCORE, which a number of people picked up. It concerned the DATA in lines 130 & 138. I have added a few enhancements as well, see lines 175/180/630/685/720/745; the revised listing is:-

```
] LIST
72  PRINT  CHR$ (18): PR£ 0: IN£
    0: CALL 1002
100  DIM NAME$(105)
110  TEXT : HOME
120 D$ =  CHR$ (4)
130  FOR N = 8192 TO 8223: READ M
    : POKE N,M: NEXT
132  DATA  169,32,160,10,32,181,1
    83,96
134  DATA  0,0,1,96,1,0,17,0
136  DATA  27,32,0,33,0,0,1,0
138  DATA    0,96,1,0,1,239,216,
    0
140  HTAB 15: PRINT "CATALABEL"
150  HTAB 8: PRINT "CATALOG LABEL
     GENERATOR"
```

```
160 ISECT = 8207:ICMD = 8214:IBUF
    P = 8211
170  REM ***READ CAT INTO MEM***
175  TEXT : HOME : VTAB 10: PRINT
    "ENTER TITLE OF DISC  > ";: INPUT
    TT$
180  VTAB 10: CALL  - 868: PRINT
    "SET UP PRINTER AND SWITCH O
    N-THEN-": PRINT "PUT DISC IN
     DRIVE AND PRESS RETURN": GET
    A$: IF A$ <  >  CHR$ (13) THEN
    180
190  TEXT : HOME : VTAB 10: HTAB
    13: PRINT "GETTING CATALOG"
200 CMD = 1:BUFP = 33:SECT = 15: POKE
    ICMD,CMD
210  POKE ISECT,SECT: POKE IBUFP,
    BUFP
220  CALL 8192
230 SECT = SECT - 1:BUFP = BUFP +
    1: IF SECT >  = 1 THEN 210
240  TEXT : HOME : VTAB 10: HTAB
    13: PRINT "SORTING CATALOG O
    UT"
250  VTAB 20
260  REM  *** FILE INPUT ROUTINE
    FROM MEM TO ARRAY***
270 B = 8459
280 N = 0
290  FOR R = 0 TO 14
300  PRINT ">";
310 A = B + 256 * R
320  FOR S = 0 TO 6
330  PRINT ",";
340 N = N + 1:NAME$(N) = " "
350  IF  PEEK (A + S * 35) = 255 THEN
    N = N - 1: GOTO 570: REM  **
    * IF FILE DELETED THEN IGNOR
    E***
360 TYPE =  PEEK (A + 2 + S * 35)

370  IF TYPE < 128 THEN NAME$(N) =
    NAME$(N) + " ": GOTO 390
380 TYPE = TYPE - 128:NAME$(N) =
    NAME$(N) + "*"
390  IF TYPE = 0 THEN NAME$(N) =
    NAME$(N) + " T"
400  IF TYPE = 1 THEN NAME$(N) =
    NAME$(N) + " I"
410  IF TYPE = 2 THEN NAME$(N) =
    NAME$(N) + " A"
420  IF TYPE = 4 THEN NAME$(N) =
    NAME$(N) + " B"
430  IF TYPE = 8 THEN NAME$(N) =
    NAME$(N) + " S"
440  IF TYPE = 16 THEN NAME$(N) =
    NAME$(N) + " R"
450  IF TYPE = 32 THEN NAME$(N) =
    NAME$(N) + " B"
460 LGTH =  PEEK (A + 33 + S * 35
    )
470 LH$ =  STR$ (LGTH):L =  LEN (
    LH$)
480  FOR W = 0 TO 2 - L:LH$ = "0"
     + LH$: NEXT
490 NAME$(N) = NAME$(N) + " " + L
    H$ + " "
500  FOR M = 0 TO 30
510 E =  PEEK (A + 3 + S * 35 + M
    )
520 F =  PEEK (A + 4 + S * 35 + M
    )
530  IF E = 160 AND F = 160 THEN
    570
540  IF E = 0 AND F = 0 THEN NAME
    $(N) = " ": GOTO 600
550 NAME$(N) = NAME$(N) +  CHR$ (
    E)
560  NEXT M
570  NEXT S
580  NEXT R
590  PRINT
600  TEXT : HOME : VTAB 10: HTAB
    10: PRINT "SENDING CATALOG T
    O PRINTER"
610  PR£ 1: PRINT  CHR$ (0): PRINT
     CHR$ (0)
620  POKE 1657,100
630  PRINT  CHR$ (14);" CATALOG :
     ";TT$: PRINT  CHR$ (15)
640 N = N - 1
650 BD = N / 2 -  INT (N / 2 + .0
    02): IF BD > .1 THEN P1 = 1:
     GOTO 670: REM  ***ODD NO OF
     FILES***
660 P1 = 0: REM *** EVEN NO OF FI
    LES***
670  REM ***ACTUALLY PRINTOUT***
680  FOR Z = 1 TO N / 2
685 T =  LEN (NAME$(Z)): REM ***
    TABBING CONSTANT ON FIRST FI
    ELD ***
690  PRINT NAME$(Z); TAB( 80 - T)
    ;NAME$(Z +  INT (N / 2) + P1
    )
700  NEXT
710  FOR M = 1 TO 5: PRINT : NEXT

720  PRINT  CHR$ (18): PR£ 0: IN£
    0: CALL 1002
730  TEXT : HOME : VTAB 10: CALL
     - 868: INPUT "DO YOU HAVE A
    NY MORE TO CATALOG?";AN$
740  IF AN$ = "YES" OR AN$ = "Y" THEN
    160
745  POKE 1657,40: REM  ***RESET
    PRINTER -JUST IN CASE-IF NOT
     CONTINUING WITH PROG. ***
750  END
]
```

Verbatim® Datalife™

disks for the performance of a lifetime

MINIDISKS

Verbatim, the world's leading producer of minidisks, makes the widest range of minidisks available for word processing and data processing applications. Intent on maintaining this position of leadership, we constantly strive to make our minidisks even better.

Each Verbatim Datalife™ minidisk is equipped with a hub reinforcing ring to aid in registration, reduce errors and give better alignment repeatability. Our dual-sided, double density Datalife™ minidisks offer users higher storage capacity. Our new plastic box makes our minidisks easier to use, safer and more convenient to store. Our testing standards go beyond the industry standard because we insist on Verbatim being the industry standard of excellence.

Introducing

CLEANING DISKS

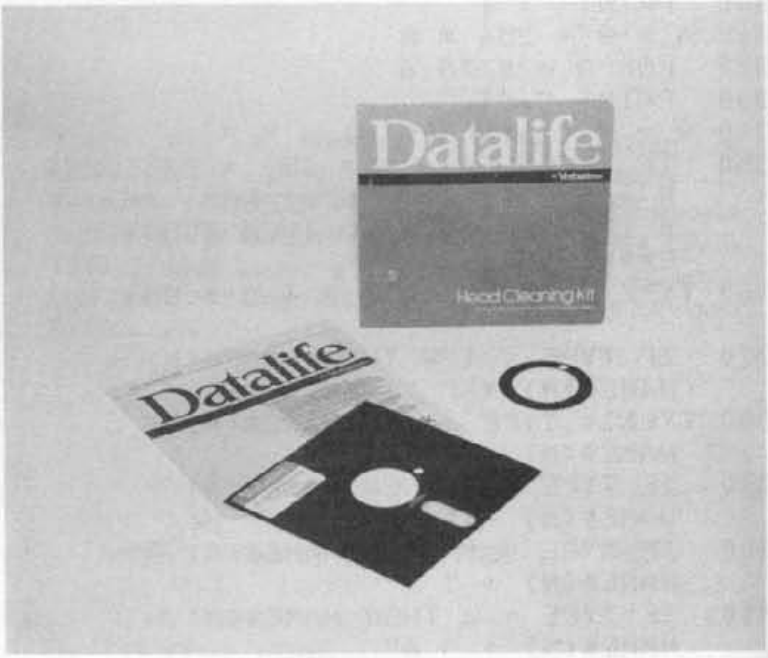
The Verbatim·Datalife™ Head Cleaning Disk is a convenient, reliable, safe means of removing up to 90 percent of debris contaminating magnetic recording heads used in computer and word processing systems.

The Verbatim Datalife™ Head Cleaning Kit consists of a durable, reusable, Lexan® jacket and presaturated, disposable cleaning disks.

The Verbatim approach is superior to and less expensive than other cleaning disks presently available.

The kit, available in 8" and 5¼" sizes, differs from those already available in the marketplace in the following ways:

- The Verbatim cleaning disks are disposable. This approach allows a fresh clean disk to be used each time. No accumulated debris from previous cleanings comes in contact with the heads.
- Each cleaning disk is presaturated with a precise amount of cleaning solution. Pre-saturation eliminates the inconvenience associated with splashing solvents onto a disk.



SPECIAL OFFER TO B.A.S.U.G. MEMBERS ONLY

10 x MINIDISKETTES **£19·50** INCLUSIVE OF V.A.T. AND POST.

I ENCLOSE A CHEQUE MADE PAYABLE TO B.A.S.U.G. CHEQUE No ..... PLEASE FORWARD TO ME BY RETURN AT THE ENCLOSED ADDRESS.

NAME .....

ADDRESS .....

..... POST CODE

NOW YOU SEE IT.....

8, The Fairfield
Farnham
Surrey.
6-11-81

Dear Sirs,

A week ago I boasted to a biologist at my college that was enquring about a program to measure reaction time that it was easy!! He said prove it, so I sat down yesterday evening and wrote one for the APPLE II which I enclose:-

```
0 REM REACTION TIME
5 HIMEM:24575
10 TEXT:HOME
20 START = 24576
30 COUNT = 24832
40 GOSUB 230
50 POKE COUNT,0:POKE COUNT+1,0
60 X = 3000
70 VTAB 10: PRINT "WHEN THE SQUARE APPEARS
PRESS ANY KEY"
80 FOR PAUSE = 1 TO X:NEXT PAUSE
90 GR:COLOR = 15
100 X = 2500 * RND(1):X = X + 500
110 FOR PAUSE = 1 TO X:NEXT PAUSE
120 PLOT 20,20
130 CALL START
140 POKE -16368,0
150 TEXT:HOME
160 VTAB 10
170 IF PEEK(COUNT)=0 AND PEEK(COUNT+1)=0 THEN
PRINT "TOO SLOW!!":GOTO 190
180 PRINT ( PEEK (COUNT + 1) * 256
+PEEK(COUNT)) * 23E - 6;" SECONDS."
190 PRINT:PRINT: INPUT "AGAIN (Y/N)";A$
200 HOME
210 IF LEFT$(A$,1)="Y" THEN 50
220 END
230 DATA
44,16,192,238,0,97,208,11,238,1,97,240,5,44,0,
192,16,241,96,234,234,208,246
240 FOR I = START TO START + 22
250 READ X: POKE I,X
260 NEXT I
270 RETURN
```

Yours faithfully,

Peter Wicks.

RANDOM TILING

by R.A.Fairthorne

Making patterns from repeated modules is a basic technique well suited to the computer. The modules, or 'motifs', can be chosen by rule or at random. The resulting patterns are of interest mathematically, and statistically, and are sometimes pleasing as well. Computer graphics experts have used this technique for quite a time.

Here is a simple minded program to suggest possibilities.It simulates the laying of square tiles whose motifs and internal colouring are chosen at random.To preserve unity in the randomized patterns there are four sets of motifs, each appropriate to a common theme.

The completed patterns can be stretched vertically or horizontally. This does not affect the number of plotted points in the original tiles, only the distances between them and the number of tiles in the pattern. The smaller the scale the longer it takes, and the worse the resolution.

On the other hand too large a difference between the horizontal and vertical scales may turn slanted lines into almost vertical or almost horizontal ones. Apple hi-res does not take kindly to these except in long lengths. For this and other reasons it is best to keep to horizontals and verticals and slopes of 1-to-1 (as in 'Chinese'), 2-to-1 (as in 'Islamic'), and 3-to-1 (as in 'Kells') or in combination (as in 'Star').

Also, to avoid visual chaos, it is better to restrict colours. Here you have the choice of white/orange or white/green. Colours are fixed for 'Islamic' at white/green. They can be fixed for other themes by replacing the randomizing instructions for 'A'and'B' in line 125 by specific Hcolor numbers.

Subroutines are used to define the motifs, though shape tables are the more powerful in general. This is to make it easy for the user to modify the program experimentally. The arrays dimensioned in lines 10 and 15 give the coordinates of the points to be joined in each motif, each row representing one motif. for example 'DIM C(2,7)' indicates that this theme uses 3 motifs (0,1,2) and 4 points each needing 2 coordinates; i.e 8 data items for each motif. These data are given in line 2100.

The subroutines tell how the points are to be joined, and the colours of the different paths of a motif. Thus line 500 is a subroutine for a single path joining four points (if the coordinates of the first and last points were the same it would draw a triangle). Line 600 uses this twice as a subsubroutine because 'Islamic' motifs consist of two joins of four points. Subroutine 700 joins three points and two points. Subroutine 800 joins two pairs of points.

The step size 'S', in line 110 and subsequently, is the side of the basic square. For 'Kells' it is 5; for the others 4.

If you wish to make patterns one at a time and retain them for saving to disk or to 'fill in', delete lines 130, 135. Then 'RUN' when you want to change. 'Filling in' a 'Star' pattern can produce interesting, if meaningless, results.

```
0   TEXT : HOME
5   VTAB 4: PRINT  TAB( 10)"RANDOM
      TILING": PRINT
10   PRINT  TAB( 12)"R.A.FAIRTHORN
     E": PRINT : PRINT  TAB( 15)"
     FEB. 1982"
15   FOR R = 0 TO 1000: NEXT R
20   DIM K(7,7)
25   DIM S(3,9): DIM C(2,7)
30   FOR I = 0 TO 7: FOR J = 0 TO
     7
35   READ K(I,J): NEXT J: NEXT I
40   FOR I = 0 TO 3: FOR J = 0 TO
     9
45   READ S(I,J): NEXT J: NEXT I
50   FOR I = 0 TO 2: FOR J = 0 TO
     7
55   READ C(I,J): NEXT J: NEXT I
60   TEXT : HOME : VTAB 4
65   PRINT  TAB( 17)"THEMES": PRINT
     : PRINT
70   PRINT  TAB( 12)"1. KELLS": PRINT
75   PRINT  TAB( 12)"2. ISLAMIC": PRINT
80   PRINT  TAB( 12)"3. STAR": PRINT
85   PRINT  TAB( 12)"4. CHINESE": PRINT
90   PRINT : PRINT  TAB( 18)"5. EN
     D PROGRAM"
95   PRINT : PRINT : PRINT  TAB( 1
     5): INPUT "WHICH(1-5)?";T
100  HOME : VTAB 4
105  IF T = 5 GOTO 210
110  PRINT  TAB( 10): INPUT "WIDT
     H OF TILE(1-7)?";W
115  PRINT  TAB( 10): INPUT "HEIG
     HT OF TILE(1-7)?";H
120  IF T = 2 GOTO 145
125  PRINT  TAB( 2): INPUT "WHITE
     /GREEN(0) OR WHITE/ORANGE(1)
     ?";C
130  VTAB 20: PRINT "PRESS ANY KE
     Y TO CONTINUE, SPACE BAR FOR
      MENU": GET A$
135  IF A$ = " " THEN  GOTO 60
140  GOTO 145
145  HGR : POKE  - 16302,0: REM *
     **FULL SCREEN ON PAGE ONE***
150  REM  ***TILE LAYING***
155 S = 4: IF T = 1 THEN S = 5
160  FOR M = 1 TO 241 STEP S * W
```

```
165  FOR N = 0 TO 165 STEP S * H
170 A = 2 *  INT ( RND (1) + .5) +
    2 * C + 1:B = 2 *  INT ( RND
    (1) + .5) + 2 * C + 1: REM *
    **SELECTION OF MOTIF COLOURS
    ***
175  REM ***MOTIF SELECTION***
180 P =  INT (4 *  RND (1))
185  IF T = 1 THEN  GOSUB 500: REM
    ***KELLS***
190  IF T = 2 THEN P =  INT (2 *
     RND (1)) + 4: GOSUB 600: REM
    ***ISLAMIC***
195  IF T = 3 THEN  GOSUB 700: REM
    ***STAR***
200  IF T = 4 THEN P =  INT (3 *
     RND (1)): GOSUB 800
205  NEXT N: NEXT M: GOTO 130
210  HOME : END
498  REM ***SUBROUTINES FOR MOTIF
    S**KELLS***
500  HCOLOR= A: HPLOT M + K(P,0) *
    W,N + K(P,1) * H TO M + K(P,
    2) * W,N + K(P,3) * H TO M +
    K(P,4) * W,N + K(P,5) * H TO
    M + K(P,4) * W,N + K(P,5) *
    H TO M + K(P,6) * W,N + K(P,
    7) * H: RETURN
600 A = 3: GOSUB 500
610 A = 1:P = P + 2: GOSUB 500: RETURN
700  HCOLOR= A: HPLOT M + S(P,0) *
    W,N + S(P,1) * H TO M + S(P,
    2) * W,N + S(P,3) * H
710  HCOLOR= B: HPLOT M + S(P,4) *
    W,N + S(P,5) * H TO M + S(P,
    6) * W,N + S(P,7) * H TO M +
    S(P,8) * W,N + S(P,9) * H: RETURN
800  HCOLOR= A: HPLOT M + C(P,0) *
    W,N + C(P,1) * H TO M + C(P,
    2) * W,N + C(P,3) * H
810  HCOLOR= B: HPLOT M + C(P,4) *
    W,N + C(P,5) * H TO M + C(P,
    6) * W,N + C(P,7) * H: RETURN
1800  DATA  0,0,3,1,2,4,5,5,0,0,1
    ,3,4,2,5,5,5,0,2,1,3,4,0,5,5
    ,0,4,3,1,2,0,5
1900  DATA  0,0,0,1,4,3,4,4,0,4,0
    ,3,4,1,4,0,0,4,1,4,3,0,4,0,0
    ,0,1,0,3,4,4,4
2000  DATA  0,2,2,0,2,4,1,1,4,2,2
    ,0,4,2,0,2,3,1,2,4,4,2,2,4,2
    ,0,3,3,0,2,2,4,0,2,2,0,1,3,4
    ,2
2100  DATA  0,2,2,0,2,4,4,2,0,2,4
    ,2,2,0,2,4,0,2,2,4,2,0,4,2

3000  REM  >>>>>>>><<<<<<<<
           >              <
           > R.A.FAIRTHORNE <
           >              <
           >   FEB. 1982   <
           >              <
           >>>>>>>><<<<<<<<
]
```

INTEGER BASIC

by John Sharp

A number of members have written in with questions or problems about INTEGER BASIC. In the touching hope of cutting downon some of my work, here is a summary of the answers which will tell you all youwant to know about the problem.

One of the most frequent problems is "I can't get some of the Integer programs to run. They just hang or go into monitor." This usually happens because the program overwrites INTEGER BASIC in RAM.
The second most popular query is that the Integer Basic on the DOS 3.3 MASTER does not work.

First let me say something about loading INTEGER BASIC into RAM, and secondly about how Integer Basic loads a program in contrast to APPLESOFT.

On the DOS 3.3 MASTER which you should have received with your disk drive, or conversion kit, is a machine code program called INT BASIC. When you boot the disk the HELLO program looks to see if there is a RAM CARD (alias a Language Card) in slot zero. If there is it loads the program into the card. If there isn't, it does nothing. The program is meant to run only on the language card; if you try running it elsewhere, it will not work. Trying to run it elsewhere means attempting to BRUN INT BASIC. It is possible to BLOAD INT BASIC, and you will find it will load at a location 1000 (hex). However, since it has been written to run at the very top of memory, (i.e. on the language card) it will not run. This would be like renumbering a BASIC program without renumbering all the GOSUBS, and GOTOs etc.

So you haven't got a language card, what do you do. Well, on the BASUG INTRODUCTORY DISK is a binary program called INTEGER. If you BRUN INTEGER, when it has loaded it will then come up with the INTEGER BASIC prompt ">" and you can write in Integer or load a program from disk or tape. If you look at where the program has loaded (see Beginners Page in HARDCORE last year), you will find that it too is loaded at 1000(hex). However, it has been relocated to run there and all the 'Gosubs' are correct now. But, 1000(hex) sits in the middle of memory and this is where problems arise.

To see what the problem is let us look at where programs are loaded. An APPLESOFT program loads at 801(hex) and then 802 etc, unless of course you tell it to load somewhere else (e.g. above the HI-RES pages) by Poking or otherwise altering the start of program pointers. An INTEGER program is different. It starts at HI-MEM and loads BACKWARDS. The end of the program is loaded first. Now if the INTEGER BASIC is in ROM, or on the language card then there should be no snags because the program is always below the INTEGER BASIC. BUT, if the introductory disk Integer Basic is used, AND the program is long, they collide. The actual program loads OK, but wipes out the tail end of the Integer Basic interpreter and although all may seem all right initially, at some stage, the Integer interpreter needs to use part of the code that has been wiped out. So what happens? - it bombs into Monitor.

Where do you go from there? On disk 11 in the library are a number of INTEGER BASICS which run at all sorts of places- a sort of Integer for allcomers. They are there for tape users as well, but I will come back to them in a moment. If you cannot find one suitable there, then you are just going to have to buy an Integer card or a RAM card. Collision problems may be avoided, but you might still run out of memory because you are using up a sizeable portion of RAM if you are not using the INTEGER on a RAM or ROM card.

So, that answers the problems above. There is one more. If you have the INTEGER BASIC PROMPT ">" and you type "FP" you will be placed back in APPLESOFT and the APPLESOFT "]" prompt will return.
If you now type "INT" the following will happen. If you have a ROM card (INTEGER in ROM) then the Integer prompt will return, because DOS knows it is there and switches it on. If you have a language card AND it is loaded with Integer, the same thing will happen because as far as your APPLE is concerned they are the same thing at this stage. I say switch, because this is precisely what it does. The same memory locations are used and so the APPLE has to switch the APPLESOFT ROMS off and turn on the language card.
If you have neither ROM nor RAM card then the message "LANGUAGE NOT AVAILABLE" will come up. Why, you may ask, when I just had it working. Answer! DOS has looked up to see if you have had a ROM or RAM card with INTEGER BASIC on it; since you haven't got one it says so. You can either rerun it from disk, or you can call the start of the program (CALL 1003 in the case of the Introductory disk version).

There is one other INTEGER BASIC, David Bolton's RELOCATED INTEGER, which as a disk version, has had the DOS modified so that when you do type "INT" it starts up the RAM version of Integer instead of looking to see if there is a card there.

If you are a tape user, on your Introductory Tapes is an APPLESOFT program which sets up INTEGER BASIC. You should have this running with the INTEGER prompt up BEFORE you try to load an INTEGER BASIC PROGRAM, otherwise it will not load. These programs are numbers 16-18 on the fourth side of the tapes.

Why use INTEGER at all? This is yet another question. Firstly, it is a faster language, because it does not have to do so much work with numbers, in sorting out floating points. Secondly, there are differences such as a MOD function which can be useful in certain mathematical/logical handling of data. In other respects it is not really as useful. There is no string handling (with LEFT$, MID$ etc); you have to dimension all strings; you have to be careful because all commands on a line are obeyed even if there is a not true condition in an IF... THEN statement. There are a few more minor drawbacks, which means that only the diehards use it these days.

Finally INTEGER is spelt just so and not INTERGER - a truly horrible looking corruption.

SO WHAT'S A PSEUDO-OPCODE?

by Ian Trackman

The last issue of Hard Core included a number of articles containing listings of programs and/or routines written in 6502 machine code. Basic-only programmers might have had a problem since noted none of the articles - my own included - explained to the reader exactly how to enter the programs into his/her Apple. In justification, may I make three points. The articles are just that - magazine articles - written for computer-literate BASUG members and are not fail-safe instruction manuals designed for first-time users. Secondly, the authors - if I am typical - want to put their effort into the main content of the article instead of writing a series of loading and saving instructions. Finally, BASUG is now running introductory courses on machine code programming.

Nevertheless, having said that, I have been asked to write this article. Its purpose is not to teach machine code programming, but to explain how to read, enter and save machine code programs published in Hard Core.

The 6502 micro-processor at the heart of the Apple responds to a series of on/off electrical pulses. These cause it to carry out the functions, such as addition, subtraction and comparison, of which it is capable. The code for each 6502 operation consists of 8 bits, making up one byte, and is referred to as an "opcode" (short for "operation code").

To carry out an operation such as "Add 15 to 20 and save the result in memory location $340", we need to give the 6502 two other types of information, numerical values (15 and 20) and memory locations ($340). By using the appropriate opcode, we can tell the 6502 that the following one or two bytes contain data - a value or an address - relating to that opcode, rather than another instruction. It's somewhat similar to a Basic command such as GET, which must be followed by a variable name, or PEEK, which must be followed by a memory address.

For the convenience of programmers, the binary opcodes and data are expressed in hexadecimal (base 16) notation. If you enter the Apple's Monitor with a CALL -151 and then type F800.F87F <return>, part of the Monitor's opcodes and data will be displayed in hex. This is known as a "hex dump". Each line starts with the number of the first address, followed by the hex values of the contents of eight consecutive bytes of memory. There is no indication in the dump itself as to which bytes are opcodes and which are data. To find that out, you would need to refer to a table of 6502 instructions and decode each number, seeing whether it needs to be followed by none, one or two bytes of data.

When we program in Basic, we hardly need concern ourselves with "memory management", that is, where exactly in the computer's memory the program and its variables will be stored. The Basic interpreter takes care of that for us. In machine code programming, there is no such luxury. The programmer has to decide where to store the program and its accompanying data, making sure that it won't be incompatible with anything else stored in memory. So we need a starting address or "origin" for our program.

T. Tse's Nicelister on pages 26 and 27 of Hard Core for April 1982 is presented in the form of a hex dump. Its origin address is the first address of the listing, 8000. Being machine code, this is $8000 (hex) and not 8000 (decimal). That's an occasional trap for the unwary - location AA60 is obviously hex, but what about 800 ? If you aren't certain, convert the origin into both hex and decimal (see Hard Core of August 1981 for Hex / Dec converters). Then look at the memory maps in the Apple Reference Manual to see which number - hex or decimal - looks more sensible.

To enter Nicelister, the first step is to give the Monitor the starting address of the program. Assuming that you are still in the Monitor, type :-

8000 : <no return>

As you are in the Monitor, don't prefix the number with a $. If you do, the Monitor will reject it, after the next <return>, with a beep. (Not even a friendly "Syntax Error" message).

Now start key-bashing. Type in each hex number, separated from the next by a space. Single numbers (00, 0D, 09, etc.) can by typed as 0, D and 9. As you can enter up to 255 characters (i.e. key-presses including spaces) between <returns>, you can type in several lines of hex numbers without pressing <return> at the end of each printed line. Don't enter the address (8008, 8010, etc) at the beginning of each new line. For reasons which follow, I suggest that you enter exactly 8 lines of hex-dump at a time. As a warning against over-filling the keyboard input buffer, the Monitor will beep after 247 characters.

If you ignore the warning, you'll over-write earlier characters. After each <return>, start the next line with a colon, to tell the Monitor that you are continuing to put more bytes into memory. If you forget the colon, the Monitor will think that you are asking for a listing of the contents of memory locations and will respond after the <return> with something like :-

20- 45
F8- FC
E6- 91

and so on. If you are fast enough, press <reset> before your original typing scrolls off the screen. Now type a new origin address followed by a colon. If you enter numbers in 8-line blocks, it's simply a matter of entering the address at the start of the correct block. Then move the cursor up and copy over your original typing. If you weren't fast enough and your original numbers have disappeared, you'll just have to re-type them.

After twenty or so lines, you'll begin to realise that direct entry, or "hand-coding", is very prone to typing errors. I suggest that you check your work as you go. After every 8 lines, do your own hex dump and compare it with the printed listing. For example, to check your typing of the first 8 lines of Nicelister, type :-

8000 . 803F <return>

If you have made a mistake in one byte, change it. For instance, if you entered B6 as the third byte of the third line, type :-

8012 : D6 <return>

If you have left a byte out or added an extra one in, you can shuffle the following bytes up or down by using the Monitor's memory move routine, but the quickest way is probably to re-enter the incorrect section.

To continue, enter the address at the beginning of the next 8 lines, followed by a colon, and type the next block, until you have finally entered the whole program.

You now have to save it. DOS needs to know the origin address and the length of the program. Fortunately, DOS accepts hex numbers (preceded by a $ sign). The origin address is easy. As to the length, remember that a program starting at $800 and ending at $880 is actually $81 bytes long. Usually the subtraction sum "end address minus start address" should be easy to do, but if not, you can use the Monitor's own hex number subtraction facility. For example, if a program starts at $9D6 and ends at $B1F, type :-

1F - D6

which will give you :-

= 49

There is a "borrow" of $100. $B00 - $900 = $200, so that, taking into account the borrow and the last byte of the program, its length is $14A. You can now save the program by :-

BSAVE <file name>, A$<hex origin>, L$<hex length>

Working out hex numbers to make up a program is a very difficult way to talk to your Apple. There is a better method, called assembly language programming, in which the 6502 hexadecimal instruction codes are replaced by three-letter mnemonics. For example, the code for the hexadecimal command for subtraction is SBC. In the simplest form of assembly language programming, data (numbers and addresses) are still expressed as hex numbers. To enter a program, you need an "assembler" and to read it back you need a "disassembler".

Let's start with the disassembler. From the Monitor, type F800L. This will produce 20 lines of disassembled code. Apple's internal disassembler has worked out - from the bit-pattern of the opcodes - which bytes are instructions and which are data. Look at the first 4 lines :-

```
F800-   4A        LSR
F801-   08        PHP
F802-   20 47 F8  JSR   $F847
F805-   28        PLP
```

The first column contains the memory address, the second column contains one, two or three hex numbers, the third column contains the mnemonic opcode and the fourth column may contain data. In the above example, only the JSR ("jump to sub-routine") opcode requires an address in the fourth column. If you now compare this with the hex dump (F800.F880), you'll see that the opening addresses in the first column and the hex bytes in the second column correspond.

So, by entering the origin address (followed by a colon) and the hex bytes listed in the second column, you can now enter Les Budgen's program on page 21 of the last issue of Hard Core. This time, checking is easier. Not only should columns 1 and 2 match the printed listing but so should columns 3 and 4.

Next, we can improve the method of entering the program. Look at Les' program. We could enter it by typing :-

300 : A9 4C A2 18

and so on, as explained above. Unfortunately, we can't type :-

300 : LDA #$4C LDX #$18

etc. However, there is a way. If you have an "old" Monitor, you can use the built-in "Mini-Assembler". If you have an Auto-start ROM, you'll need a copy of the re-located Mini-Assembler from the BASUG program library. Instructions for using it are set out on pages 49 - 51 of the Reference Manual and I won't repeat them here.

When using the Mini-Assembler, it is not necessary to type the $ sign to signify hex numbers, although you must enter all # signs (which indicate real numbers) wherever they occur. You need only one space between columns and, unlike the hex-dump method of entry, you must type a <return> after each line. Other than that, entering a program through the Mini-Assembler is a case of setting the start address and copying out columns 3 and 4 of Les' listing.

The Apple disassembler tries to interpret every hex byte as part of a 6502 instruction. In reality, some programs contain variables or blocks of data which cannot be translated in that way. If the disassembler cannot perform its translation properly, it produces 3 question marks. Les' program contains such a data block and ??? appears at $314 and $316. You cannot enter these bytes through the Mini-Assembler and you have to go back to the hand-coding method. These data bytes are always single bytes in the disassembler listing. When you come across ??? in the listing, I suggest that you make a temporary entry in the Mini-Assembler at that address of the opcode BRK, which corresponds to the hex byte 00. After typing in the rest of the program, you can then alter those particular bytes to their correct values with the direct entry method.

The "Mini" in Mini-Assembler means that it can't do everything that a big assembler can do. For example, using a full assembler, we can add comments into our programs, we can refer to variables and address by giving them names (or "labels" as they are called), we can enter text as ASCII characters and we can give instructions to the assembler as to how we want it to produce listings of the program. Assemblers for the Apple offer different facilities and therefore their operating instructions (called "pseudo-opcodes" or "assembler directives") vary from one to another. This means that listings are non-standard. Nevertheless, they do have many features in common, so that it is still possible to enter programs using a combination of the Mini-Assembler and the direct hex entry method.

Look at my Apple Writer Ramcard Patcher on pages 15 to 17 of the April Hard Core. The listing has now expanded to 7 columns wide. Columns 1 and 2, separated by a colon, are the address and hex code as before. The third column is the line number within the listing. It's not used by the program itself, but merely for reference purposes when listing and editing the program.

The next column contains an address label. It's somewhat like a line number in Basic and we only use it when we need to refer to the line, such as in a "GOTO"-type command. The fifth column contains the familiar 3-letter opcode. It can also hold a pseudo-opcode, such as "ASC" in line 95, which tells the assembler that what follows is some text in ASCII.

Column 6 holds addresses, numbers, variables and data and the last column is for comments.

The program itself will usually start with a block of "equates" which assign labels to addresses and constants to be used in the body of the program. There will also be an ORG pseudo-opcode, giving the starting address of the program.

There are three ways of entering such a program. If you have the same assembler as the author, then just copy the listing. (Please see my comments about < and > signs in the DOS Toolkit Assembler in Hard Core of March 1981). If you have a different assembler, we'll assume you know enough about assembly language programming to do any necessary conversion work.

The second method is to hand-code the program. Look for the origin (ORG) and then copy the hex bytes from column 2.

The third method is to use the Mini-Assembler. Here, you'll need to translate the labels and variables back to hex numbers. The easiest way is to refer to the hex bytes in column 2. You'll have to keep in mind that in 6502 code, 2-byte addresses are always referred to in reverse order, so that the low-byte comes before the high-byte.

Let's look at my Patcher program in more detail. It starts with a title, followed a block of equates, then the ORG $9C00 statement at line 31. This tells us where to start assembling. (Remember to relocate DOS before you begin !).

The first line, LDX #0, is straight-forward and can be typed into the Mini-Assembler without trouble :-

```
300 : LDX #0
```

In line 36, we can ignore the label "GO.LOAD" for the time being. However, in column 6, we have the label "TEXT", which needs translation. ("TEXT" has nothing to do with the Basic command TEXT - it's just a name that I made up.) Look across to column 2, where you'll see the three hex bytes :-

```
BD 47 9C
```

The first one always corresponds to the opcode (LDA in this case) and the next two are the numerical equivalent of the label "TEXT". Remembering to reverse the low-high order, you can now type into the Mini-Assembler :-

LDA 9C47,X
Notice that the ",X" is not part of the label and must be copied exactly as listed.

Another way of finding the correct value for TEXT is to look down column 4 until you spot it - on line 74, in fact. Then look across to column 1 - there is the address again - $9C47.

The same idea is used in the next line (37), so that you'll be typing :-
BEQ 9C0D

The next line (38) is slightly different. COUT is not a label within the program itself, but it appears in the equates list at the top. So we type :-

JSR FDED

In line 42, Hard Core's printer was left in English character mode and so the hash signs have been printed as pounds. That line will be entered as :-

LDA #9C

When we reach line 74, we've already seen that the Mini-Assembler can't cope with text. You'll now have to go back to the old direct entry method to put in the last few bytes of the program.

If you list the program on the screen, columns 1 and 2 should correspond, although the disassembler will split up the text lines (75 - 80) differently. You should also see the similarity between the opcode and data columns on the screen and in the printed listing.

To close this article, let me add a tip about loading a machine code sub-routine from Basic. Of course, you can BLOAD it from within your Basic program. However, if it is short, you may want to POKE it in directly. There is a routine on page 77 of the DOS Manual, which creates a whole batch of POKE commands in your program. A neater way is to have a loop, such as :-

10 FOR I = 768 to 780
20 READ X
30 POKE I,X
40 NEXT
50 DATA 43,56...(the decimal values of the program)

Put the machine code into memory in one of the ways explained above, then write lines 10 to 40 in the normal way. Now enter this immediate command :-
PRINT "50 DATA"; : FOR I = 768 TO 780 : PRINT PEEK(I) ","; : NEXT
The next line of your Basic program will appear on the screen ready for you to move the copy cursor over it. (Don't enter the last comma.)

As I said at the beginning of this article, I have not attempted to explain what the opcodes and pseudo-codes do. Perhaps you'll now be tempted to find out !

USING DOS FROM THE MONITOR AND IN MACHINE LANGUAGE

By John Kleeman

The following notes refer to DOS 3.2 on an ITT 2020 with 48K. All numbers are in hex.

To use DOS commands from terminal in monitor (e.g.CATALOG)
3EAG or A851G hooks in DOS (e.g.after RESET)
CALL -151 enters monitor for Basic
It is also necessary to (or at least it works if you do so) 76:FF (i.e.put FF into location 76)

To trap DOS errors:

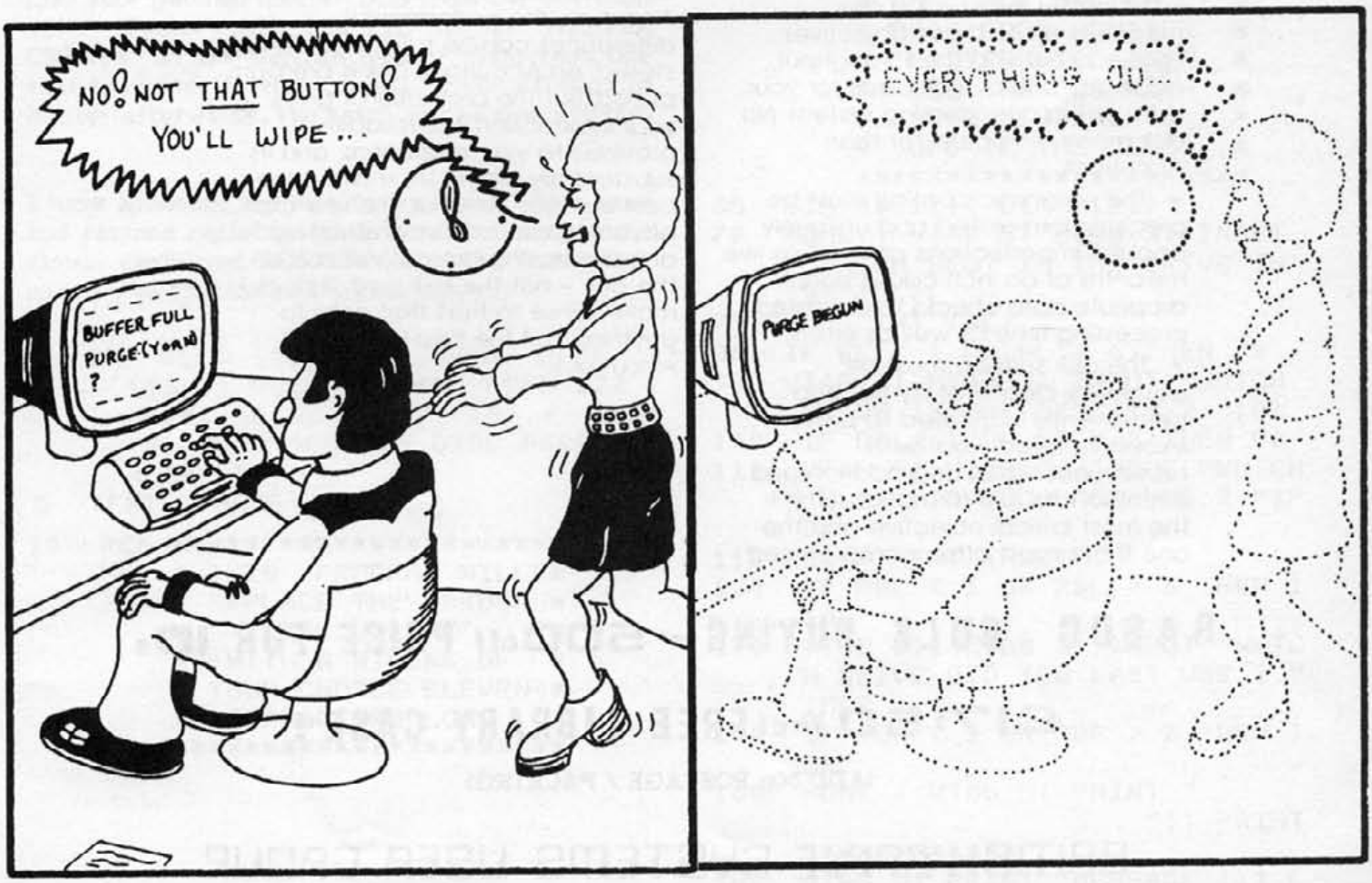
Put address of error-trapping routine in to lo-hi format into locations 905A,905C,905E

The error trapping routine will then be called with the DOS error code as per BASIC ONERR will be in the X-register.

To suppress DOS error messages:
Overwrite A6D8:F015

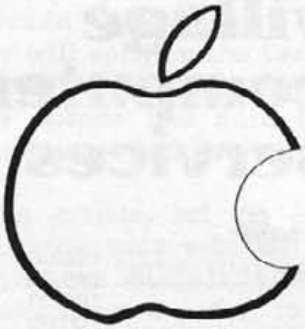
To use DOS in machine-code programs is similar to Basic- you just preface DOS commands with control D. Problems arise with text-files; these can be got round by 76:0
33:8F (this is putting the character '?'in the monitor prompt register)

The above works through I am not always quite sure why, in particular the use of location 76 (hex) is obscure.



Accutrack Disks . . .

Because data reliability is the important difference in disk construction.



Anatomy of a disk.

Flexible disks are simple information storage devices consisting of a magnetic disk enclosed in a semi-stiff protective jacket. The disk rotates within the jacket while magnetic recording heads on your data or word processing systems "read" or "write" information on the disk's magnetic surface. Since disk operation is simple, it's relatively easy to make one that works. But building in reliability is something else again. It takes specialized technology to build disks that operate flawlessly over an extended period of time.

What counts in disk construction.

Key design objectives for a disk are listed below. How well a disk measures up to these objectives relates directly to the throughput, accuracy and overall costs for your data or word processing system. No disk measures up better than Accutrack.

- The magnetic coating must be precisely formulated and uniformly applied. Imperfections as small as five millionths of an inch cause signal dropouts, data checks and wasted processing time as well as errors.
- The disk surface must be absolutely clean, totally flat and permanently lubricated to prevent excessive head wear with subsequent signal degradation and eventual loss of information. (This is the most critical objective and the one that's most often compromised since poor operating results take a while to show up. It's also the area that most effects the long term reliability of your data.)
- The disk must be free to rotate within its jacket without internal drag to avoid further data checks, excessive processing times and errors.
- The jacket must protect the disk from external contamination and damage. It should also remove microscopic particles of debris from the disk surface before they can damage the disk.

Why you'll never find the best disk bargain in the bargain basement.

While there's little apparent difference between other disks and Accutrak, the performance differences can be substantial. Simply stated, an Accutrack disk is premium priced. But the protection it gives your information; the reliability it provides to your operations; and its substantially longer life make it the best disk buy. After all, the real cost of your operations is constructing and processing the data stored on the disk – not the disk itself. It doesn't make sense to trust that data to anything but the best disk. Accutrack.

BASUG BULK BUYING – 50% off PRICE FOR 10s

£17:50 incl FREE LIBRARY CASE !

(ADD 50p POSTAGE / PACKING)

BRITISH APPLE SYSTEMS USER GROUP

P.O. Box 174, Watford WD2 6NF

PERSONALISED DISKS

by Rex M.F.Smith

DOS 3.3 disk personalizer will add an eleven character string of your choice to Track #2, Sector #2 at Hex $AF. This message replaces 'DISK VOLUME' and is displayed every time you boot the disk. The idea came from Windfall January, 1982, and the information on how to patch the disk came from 'Beneath Apple DOS'. The routine will relocate itself at $3000 and load the sector at $2000. I chose these locations as the routine to call RWTS can then be used as in the DOS manual, page 94ff. It is probably not a good example of how to use RWTS. It also changes the DOS RAM image of the sector as i found that DOS does not read the information off the disk, but supplies 'disk volume' from RAM at 46011 decimal. Now when you call the catalog up, your message will appear (for each disk until you reboot with a different message). The programme exits by calling the catalog and thus displaying your message.

If you wish to patch a master disk in this way, master the disk first and then use the routine. To add the patch to other masters, BLOAD 'MASTER CREATE' and change the disk for the master patched, then enter the monitor and type 800g to execute 'MASTER CREATE' which will load your patched master DOS from the switched disk. Easier still, use FID to put 'MASTER CREATE' on the patched master disk, then use that to master further copies. If you attempt to master afterwards, the patch will be overwritten.

I have appended the monitor routines which save and restore registers as I found the programme slowly destroyed DOS without them. This patch appears to have cured the problem.

```
1   IF  PEEK (104) = 8 THEN  POKE
    103,0: POKE 104,48: POKE 122
    88,0: PRINT  CHR$ (13) +  CHR$
    (4);"RUN DOS 3.3 DISC PERSON
    ALIZER"
5   TEXT : HOME
10  REM ************************
        * THIS  PROGRAM WILL *
        * REPLACE THE WORDS  *
        *   'DISK VOLUME'    *
        *  WITH A STRING OF  *
        * YOUR CHOICE ELEVEN *
        *  CHARACTERS LONG   *
        ************************
20  REM ************************
        * USES THE RWTS FROM *
        * CODE STORED AT HEX *
        * LOCATION 3000 AND  *
        * STORES THE SECTOR  *
        *  AT HEX 2000-20FF  *
        ************************
30  REM ************************
        * IT CHANGES TRACK 2 *
        * SECTOR 2 BY POKING *
        *  THE USER STRING   *
        *  (BACKWARDS) INTO  *
        * HEX OFFSET $AF OF  *
        * THE SECTOR STORED  *
        ************************
40  REM ************************
        * EACH BYTE MUST BE  *
        * 128 + ASC VALUE OF *
        * THE CHARACER USED  *
        * RWTS IS THEN USED  *
        *  AND REWRITES THE  *
        *  DATA TO THE DISC  *
        ************************
50  REM ************************
        * IF YOU USE 'MASTER'*
        *  FIRST, THIS DISC  *
        * CAN BE USED BY THE *
        * MASTER CREATE PROG *
        *  BY BLOADING IT    *
        * THEN SWITCH DISCS  *
        ************************
60  REM ************************
        * AND CALL MASTER BY *
        * 800G FROM MONITOR  *
        *  MASTER WILL THEN  *
        *  USE THE DOS WITH  *
        *  YOUR MESSAGE TO   *
        *  UPDATE THE DISC   *
        ************************
65  DIM A$(11): DIM A(11)
70  HOME : VTAB 5: HTAB 3: INPUT
     "WHICH SLOT IS THE DRIVE IN
     ? ";SL
75 SL =  INT (SL)
80  IF SL < 1 OR SL > 6 THEN 70
90  VTAB 7: HTAB 3: INPUT "WHICH
     DRIVE IS THE DISC IN ? ";DR
100  IF DR < 1 OR DR > 2 THEN 90
110  VTAB 9: HTAB 3: INPUT "WHICH
     SLOT DID YOU LAST USE ? ";P
    SL
115 PSL =  INT (PSL)
120  IF PSL < 1 OR PSL > 6 THEN 1
    10
130  VTAB 11: HTAB 3: INPUT "WHIC
    H DRIVE DID YOU LAST USE ? "
    ;PDR
140  IF PDR < 1 OR PDR > 2 THEN 1
    30
160  HOME : VTAB 5: PRINT "
                          ";: PRINT
    "**********"
180  VTAB 5: PRINT "MESSAGE (11 C
    HARACTERS) ?";
```

```
190  GET A$:A$( POS (A) - 24) = A
     $: PRINT A$;: IF  POS (A) <
     36 THEN 190
200  PRINT : INPUT "IS THE STRING
      O.K. ?";Y$: IF  LEFT$ (Y$,1
     ) = "Y" THEN 220
210  FOR N = 1 TO 11:A$(N) = "": NEXT
     N: GOTO 160
220  FOR N = 1 TO 11:A(N) =  ASC
     (A$(12 - N)) + 128: NEXT N
500 ADRS = 3072: REM $0C00
520  POKE ADRS,169: POKE ADRS + 1
     ,12
540  POKE ADRS + 2,160: POKE ADRS
      + 3,10
560  POKE ADRS + 4,32: POKE ADRS +
     5,217: POKE ADRS + 6,3
580  POKE ADRS + 7,96: POKE ADRS +
     8,0
590  POKE ADRS + 9,0
600  POKE ADRS + 10,1: POKE ADRS +
     11,(SL * 16)
620  POKE ADRS + 12,DR: POKE ADRS
      + 13,0
640  POKE ADRS + 14,2: POKE ADRS +
     15,2
660  POKE ADRS + 16,32: POKE ADRS
      + 17,12
680  POKE ADRS + 18,0: POKE ADRS +
     19,32
700  POKE ADRS + 20,0: POKE ADRS +
     21,0
720  POKE ADRS + 22,1: POKE ADRS +
     23,00
740  POKE ADRS + 24,00: POKE ADRS
      + 25,(PSL * 16)
760  POKE ADRS + 26,PDR
765  POKE ADRS + 27,0: POKE ADRS +
     28,0: POKE ADRS + 29,0: POKE
     ADRS + 30,0: POKE ADRS + 31,
     0
780  POKE ADRS + 32,00: POKE ADRS
      + 33,1
800  POKE ADRS + 34,239: POKE ADR
     S + 35,216
820  CALL ADRS
840  FOR N = 1 TO 11
860  POKE 8192 + 175 + N,A(N)
870  POKE (45999 + N),A(N)
880  NEXT N
900  POKE ADRS + 22,2: POKE ADRS +
     23,0: POKE ADRS + 24,0: POKE
     ADRS + 25,(SL * 16): POKE AD
     RS + 26,DR
940  CALL ADRS
960  PRINT  CHR$ (4);"CATALOG"
970  PRINT : PRINT "DO YOU WISH T
     O ALTER ANOTHER DISC ?";
980  GET Y$: IF Y$ <  > "Y" THEN
     1000
990  RUN
1000  PRINT : PRINT : PRINT "PROG
     RAMME POINTERS ARE BEING RES
     ET"
1010  PRINT  CHR$ (4);"FP"
```

WRITE PROTECT TABS NO MORE!

By Jurgen Wolda

(first printed in USER Magazin Nr 5, 81, the magazine of AUGE, the German user group. Translated by Tony Williams)

Read on all ye who are too lazy to want to carry on with the drag of removing the write protect tabs from disks and replacing them after your priceless programs are written.

This contribution to the magazine describes a cheapo solution open to all owners of a soldering iron, a drill and a smattering of DIY skills.

Beneath the left disk guide rail in the drive is a microswitch which reads the 'write-protect' cut-out in the disk. If you break one of the wires leading to the microswitch you kid the system into believing that the cut-out has been covered and so writing to disk is no longer possible.

By now connecting this break point to a small flip switch on the front panel of the drive you can save yourself all that sticky messing about with tabs.

So much for the theory, now for the brass tacks.

The first thing to do is systematically dismantle the drive. That sounds worse than it is, since only eight screws have to be removed in all. The first four can be found on the underside of the drive. Once removed the drive cover can be slid off (to the rear!). The next four screws secure the black frontal panel to the metal chassis of the drive and are located to left and right of the panel. Unscrew these and remove the 'In Use' diode light. Now drill a hole in the panel next to or above the diode light big enough to take your switch, and fit it. Next solder two wires from 5 to 7 cm long to the switch and slip two lengths of insulating sheath over each. Now cut the wire leading to the drive's microswitch, strip both ends and solder them to the wires coming from your switch. Finally push the insulating sheath over the joins. Replace and screw down the frontal panel. After replacing the diode light the drive's housing can be slid back on and secured with the screws. End of operation.

POLISHING THE APPLE

(or how to construct and install a lower case keyboard and display unit).

by Peter Blair

The reasons.

One of the few annoyances of the Apple II is the lack of lower case and proper shift key operation. The keyboard enhancer from Videx is a possible solution but an expensive one. Other modifications seem to sacrifice some compatibility with existing programs.
The problem therefore, was to provide my Apple with full lower case cheaply and simply and preserve all the existing display features. The circuit designs which follow were the result. These have been thoroughly tested and are installed and working on my own Apple (except the pre-revision 7 display circuit). They are simple to construct and require no modification of the Apple circuit boards except the addition of a few wires. One minor point is that there are no true descenders possible because of the way the Apple structures the text screen, but this does not make a lot of difference once you get used to it.

The logical approach.

Ignoring the fact that the Apple software would convert lower case input to upper case I decided that the first step was to provide a full upper/lower case capability to the existing keyboard plus a correct shift key operation and a shift-lock key. To simplify this and avoid an extra switch I used the SHIFT and CTRL keys together to give shift-lock.

The diagram FIG.1 shows the logic circuit used. If both the SHIFT and CTRL keys are pressed together the flip-flop IC2a changes its output state. When the Q output is low the keyboard is in normal Apple upper case only mode. If the Q output is high then lower case is enabled, and the LED lights to indicate this. In this mode the circuit examines bit 6 of the keyboard output and the state of the shift key. If bit 6 is high then a letter key has been pressed. If the shift input is low then the shift key has been pressed and bit 5 is left unchanged, otherwise bit 5 is inverted and the letter becomes lower case. Since pressing the shift key changes the keyboard encoder output, you can still access every character on the keyboard except three when in lower case. These are the upper case M, N, and P since these keys give punctuation characters when shifted. A software fix for this minor irritation is given later.
Display changes.

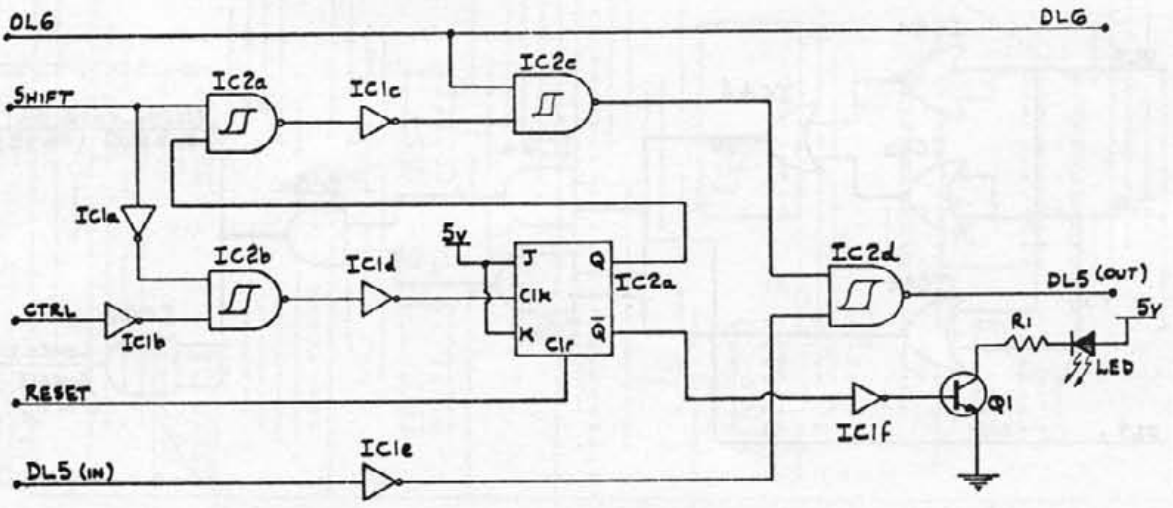


FIG.1 KEYBOARD SECTION ~ LOGIC DIAGRAM

The logic controlling the display is shown in FIG.2. The circuit detects when a lower case letter or punctuation (codes E0 to FF Hex) is being output by looking at bits 6 and 5 of the output data. Then if bit 7 is high, the logic enables a lower case character generator and disables the output of the normal character generator. If upper case is output the opposite happens. Finally, if bit 7 is low then either flashing or inverse characters are required and the normal generator is enabled. This means that it is not possible to display inverse lower case, and flashing lower case must be done with software, but gives absolute compatibility with all existing software without requiring a display change-over switch. One extra bonus is that POKE-ing control character codes to the screen produces a range of special characters.

Construction.

The parts required are given in the list at the end of the article. Having obtained these, then first you must check which version of the Apple you have. The revision 7 board uses a custom programmed character generator ROM (actually a mask programmed 2716) while the manual gives the character generator as a 2513 type. FIG.3 shows the complete vero-board layout for the revision 7 Apple. If your Apple uses the 2513 character generator on the main board then the display section of the new board should be laid out as FIG.4. Notice that the two circuits are completely independent although built on the same board for convenience.

Start by inserting the IC sockets. If you do not use sockets for the TTL then leave soldering the IC's until last, BUT YOU MUST USESOCKETS for the character generators since soldering their pins will invalidate any guarantee. Next make all the wire links using solid core insulated wire. Then insert the resistor, capacitors and transistor and solder in place. Solder connecting pins into the board for the connections marked SHIFT, CTRL, DL6, and DL7 (the last two are only required on the FIG.4 circuit), and also for the LED connections.

Using ribbon cable make up the 16 and 24 pin header plugs, and then connect the other end of the cables to the board. The cables will need to be around 18" long. Note that the plug pin numbers are given for the connections from the board to the 16 pin header, but for the 24 pin plug it is just a pin for pin correspondence between the on-board socket and the plug. Also, for the 24 pin plug, wires are only required where indicated by a small

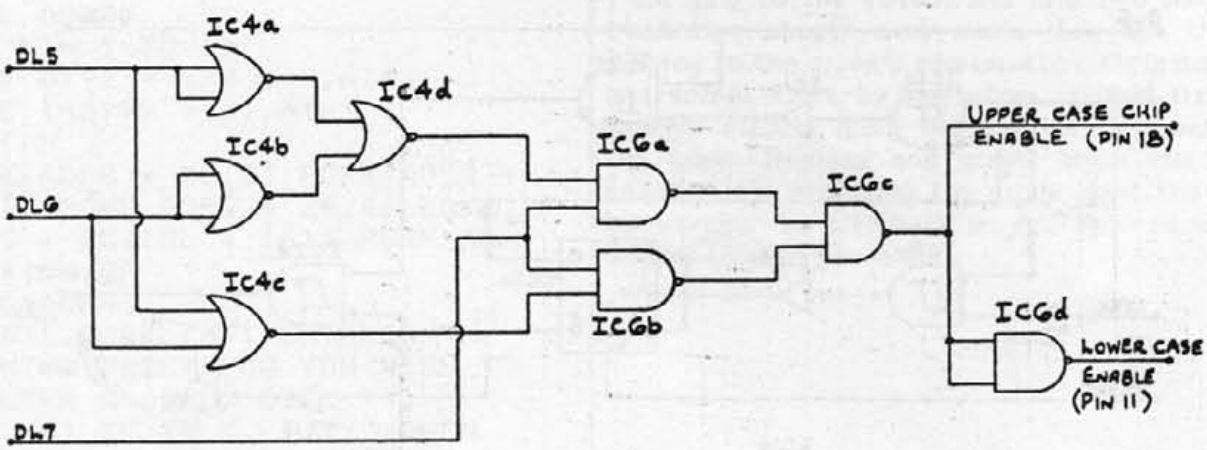


FIG 2. CASE SELECT LOGIC

circle on the diagram FIG.3 or FIG.4, in particular NO CONNECTION is to be made to pin 18 of the plug in the FIG.3 version, or to pin 11 for the FIG.4 version since these are the chip enable pins. Now make all the track breaks that are required (marked by X in the diagram) but be careful not to miss any or to break any additional ones since some connections are made by direct link between IC pins. Lastly install all the chips except IC7 which must first be removed from the main board. Ensure that all IC's have the indent loacating pin 1 to the top of the board as in the diagram.

Installation.

Disconnect the Apple from the supply and remove all peripheral cards from their slots noting where they go. Turn the Apple over onto a cloth covering your work bench. Carefully remove the four screws from the bottom front edge noting what they look like. Now remove the six screws around the outside bottom edge of the Apple. DO NOT REMOVE ANY OTHER SCREWS. Hold the top and bottom tightly together and turn the Apple right way up. Gently raise the front edge and remove the keyboard connector from its socket. The top may then be removed completely. Locate the character generator at location A5 and carefully remove it. Fit this to the new board as IC7.

The extra wires should now be attached as follows. Turn over the keyboard section of the Apple so that you are looking at the rear of the encoder board. Cut two lengths of wire about 12" long in different colour insulation. Locate the CTRL and SHIFT keys. These have a common connection which is to ground, with independent output connections. Trace these back to the plug betweeen keyboard and encoder and carefully solder one wire to each connection at the back of the encoder board. On my Apple these were pin 3 for CTRL (or you could connect at the slide switch terminal), and pin 24 for SHIFT, but you should check this first. Make certain that these additions do not loosen or short any the existing connections. For the display there are no additional connections unless you are using FIG.4. To make these proceed as follows. Locate chip B11 on the main board and carefully solder a wire to pin 12 for DL6. Similarly connect a wire to pin 6 of the chip at B13 for DL7. Note that pin 1 of all chips is indicated by a white dot on the circuit board adjacent to the chip.

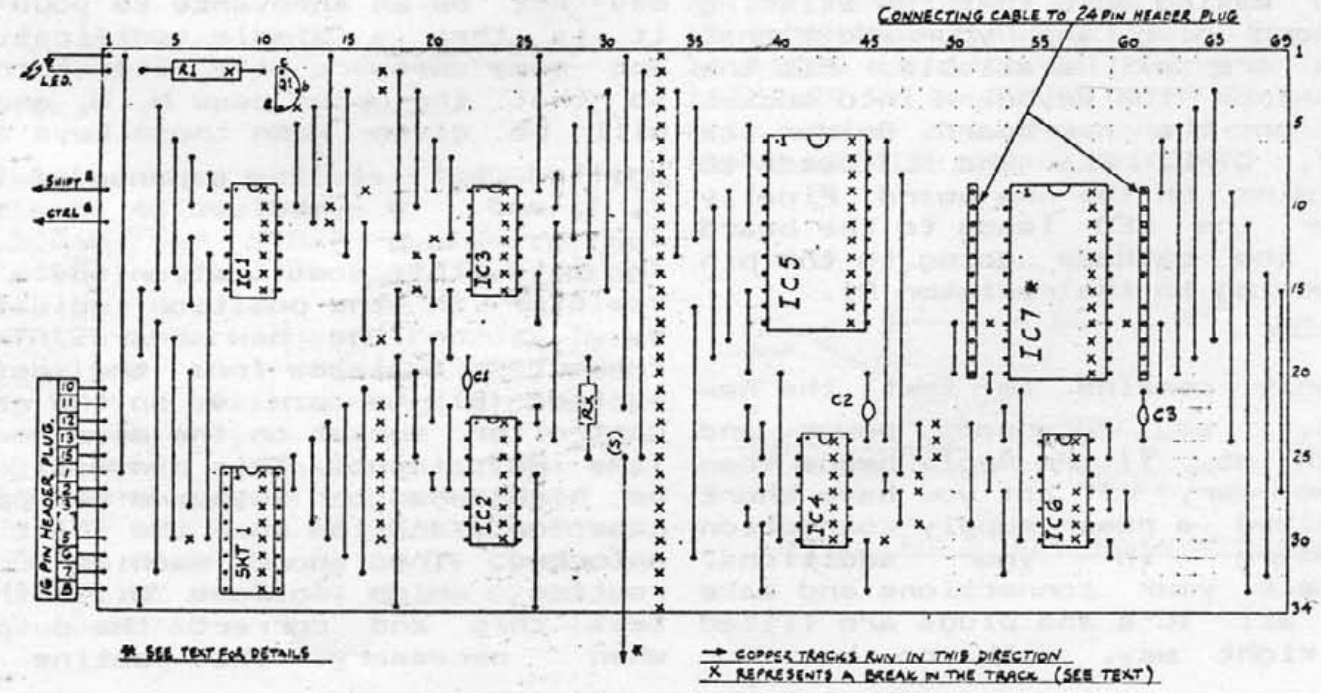


FIG.3 VERO BOARD WIRING LAYOUT (BOTH CIRCUITS)

Mount the new board on top of the power supply case (at the rear left of the Apple) with the keyboard cicuitry closest to the keyboard. To do this use a couple of double-sided sticky pads, one on top of the other, at each corner of the board. Make sure the soldering on the underside of the board does not touch the power supply case. Plug the 16 pin header into the keyboard socket at location A7 on the main board making sure pin 1 is in the right position. Plug the 24 pin header into the socket you removed the character generator from at location A5 again getting pin 1 in the right place.

The LED must be fitted now. Connect two insulated leads of different colours to the LED leads. These should be about 24" long. Note which wire connects to the cathode of the LED, which is the left hand lead when looking at the indented circle on the package. Twist the leads together but make sure they do not short. Gently prise up the cover on the power on light on the keyboard. Beneath you will see a bulb in a mounting with four pegs. Carefully break off the upper peg with a needle nose pliers. Feed the LED wires through the case top in front of the bulb and locate the LED against the bulb. This is shown in FIG.5. Press the cover back in position.

Fit the top of the case back on the Apple making sure that the existing keyboard plug and your additional wires are all available. Fit the plug from the keyboard into socket SKT1 on the new board. Solder the SHIFT, CTRL, DL6 and DL7 leads to the pins on the new board. Finally solder the LED leads to the board with the cathode going to the pin connecting to the resistor R1.

Testing.

It only remains to test the new board, so reconnect power and switch on. If the Apple beeps then OK so far, if not you have short circuited a power supply connection somewhere in your additions. Re-check your connections and make sure all IC's and plugs are fitted the right way, with no bent or damaged pins. If the Apple initialises correctly then you may test the board by typing something. It should appear that nothing has been changed at the moment and the Apple should still operate normally and respond to control characters as usual. If this is not so then again a short circuit or incorrectly fitted IC will be the problem.

To test the lower case capabilities the following two line program will suffice:

```
10 X=PEEK(-16384):IF X<127 THEN 10
20 PRINT X:POKE -16368,0:GOTO 10
```

This will display the code of any key pressed (with the high bit set). Pressing SHIFT and CTRL together and releasing them will release the shift lock and the codes will change. Pressing shift will give upper case codes but will not affect the shift lock. To lock back into upper case press SHIFT and CTRL together again.

To check the display, just poke the upper case codes (224 to 255 decimal) to the screen. Flashing and inverse modes should still work as normal.

The finishing touches.

As I stated earlier, three upper case codes are not available when the shift is unlocked. This may or may not be an annoyance to you. If it is then a simple modification and some software will alter things so that the upper case M, N, and P will be given when these keys are shifted but at the expense of the ^,], and @ symbols.

To make this modification add a 1K resistor in the position indicated by Rx on the new board. Then connect a lead from the point marked (S) to pin 4 on the game controller socket on the main board (the PB2 input). This pin will now be high when the keyboard is upper case only and low when the shift is unlocked. The short machine code routine which follows will then test this and correct the output when necessary. The routine is

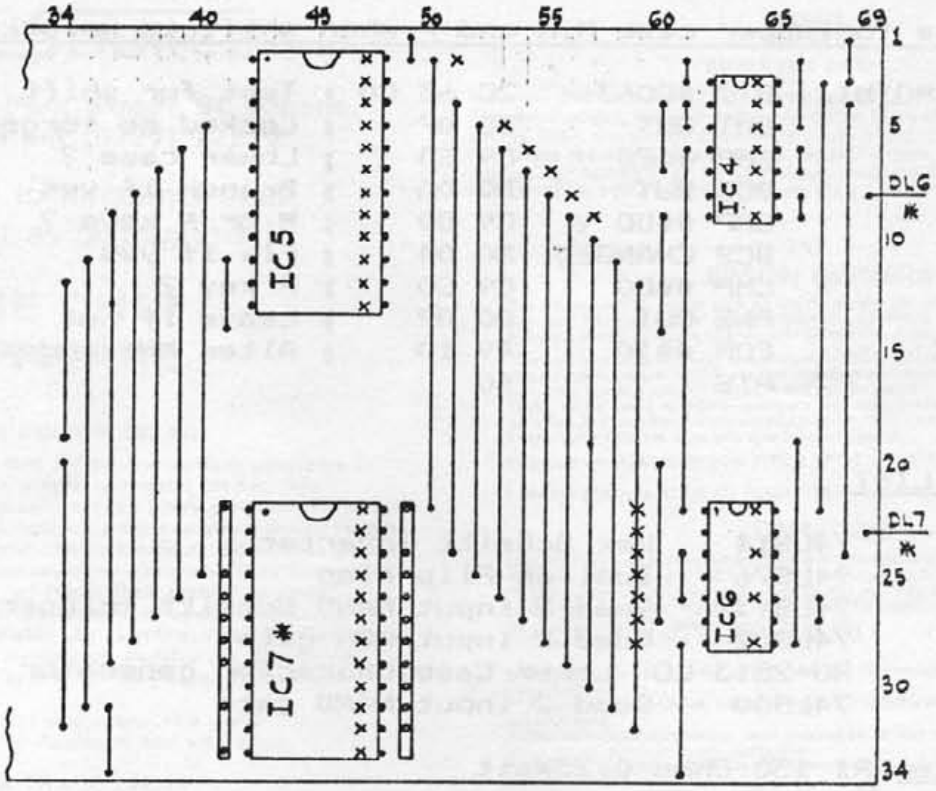


FIG 4. MODIFIED WIRING LAYOUT FOR PRE-REV.7 APPLES
* SEE TEXT

fully relocatable and should be called after reading a keypress.

Finally I have worked out patches for using this lower case modification with Applewriter and these are also given below. These incorporate the key testing routine so that all alphanumeric characters are available when the shift is unlocked.

Patches for use with Applewriter:

TEDITOR:

Overwrite the bytes at $0806 to 0808 with 4C 20 18
At $086A and $1537 overwrite the JSR $1501 with EA EA EA (ie three NOP's)
At 0A12 alter the C0 to 20 . This will allow the sequence ESC ESC CTRL C to still perform a correct case change.
Using the monitor enter the following code starting at $1820
$1820:AD 00 C0 10 FB 2C 63 C0 30 0F C9 E0 B0 0A C9 DD B0 04 C9 C0 D0 02 49 10 C9 8E D0 02 A9 DB A0 02 88 D0 FD 8D 10 C0 60

You can now save your new version of TEDITOR with BSAVE TEDITOR,A$803,L$1045

PRINTER:

At 0F56 change E1 to A1
At 15DF change E0 to A0
At 15E3 change F0 to B0
At 15E7 change FA to BA
At 15FC change F0 to B0
Replace the instructions starting at $12D8 with the following
$12D8:29 7F A6 57 F0 02 09 80 60 00
Now save the new version using BSAVE PRINTER,A$803,L$1020

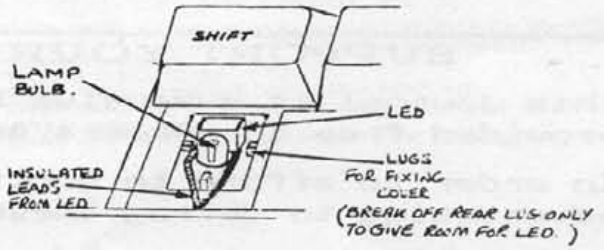


FIG 5 LED INDICATOR FITTING

Routine for upper case M,N and P when shift is unlocked.

```
ENTRY POINT.  BIT $C063     2C 63 C0 ; Test for shift lock
              BMI OUT       30 0F    ; Locked so forget it
              CMP #$E0      C9 E0    ; Lower case ?
              BCS OUT       B0 0A    ; Branch if yes
              CMP #$DD      C9 DD    ; M or N keys ?
              BCS CHANGE    B0 04    ; Fix if yes
              CMP #$C0      C9 C0    ; P key ?
              BNE OUT       D0 02    ; Leave if not
CHANGE        EOR #$10      49 10    ; Alter MNP codes
OUT           RTS           60
```

PARTS LIST

IC1	74LS14	Hex Schmitt Inverter
IC2	74LS76	Dual JK Flip Flop
IC3	74LS132	Quad 2 input NAND Schmitt trigger
IC4	74LS02	Quad 2 input NOR gate
IC5	RO-2513 LC	Lower Case Character generator
IC6	74LS00	Quad 2 input NAND gate

Resistor R1 150 Ohms 0.25Watt

Capacitors C1,C2,C3 0.1 uF 12volt working

Transistor Q1 General purpose NPN type BC548 (BC108,BC109 also suitable.)

LED Rectangular type , colour Red (Available from MAPLIN)

Plugs Solder type Header plugs. 1 required 16 pin, 1 required 24 pin.

Sockets Low profile DIL sockets 4 required 14 pin, 2 required 16 pin, 2 required 24 pin

Note that the character generator is available from TECHNOMATIC Ltd who advertise in the computing press.

You will also require a piece of vero-board (copper clad type) 69 holes x 34 tracks on 0.1" matrix. This should be available from any large electronic component suppliers.

The approximate cost of components is £15 - £16.

SHAPE-DRAW CONTINUED

In the February issue of **HARDCORE** we published Peter Cave's Shapedraw program, but unfortunately forgot to include the program listing. That omission is now corrected!

```
]LIST

10  GOSUB 4000: GOTO 1000
99  REM   PLOT CURRENT POS
100  HPLOT X - 1,Y - 1 TO X + 1,Y
     - 1 TO X + 1,Y + 1 TO X - 1
     ,Y + 1 TO X - 1,Y - 1
110  RETURN
199  REM DECTOHEX
200 N = 0:H$ = ""
210 A% = D / 16:B = A% * 16:D = D
     - B:N = N + 1
220  IF D < 10 THEN C = D + 48: GOTO
     240
230 C = D + 55
240 H$(N) =  CHR$ (C):D = A%
250  IF D THEN 210
260  FOR X = N TO 1 STEP  - 1
270 H$ = H$ + H$(X): NEXT
280  RETURN
299  REM   CALCULATE DATA AS 2 BY
     TES
300 H% = D / 256:L% = D - 256 * H
     %: RETURN
999  REM   PLOT GRID
1000  IF N = NC THEN  TEXT : HOME
     : PRINT "SHAPE TABLE COMPLET
     E WITH ";NC;" SHAPES": GOTO
     3000
1010  HGR : HCOLOR= 7:N = N + 1:T
     I = I:F = 0
1020 LR =  INT ((279 - W * 4) / 2
     ):UD =  INT ((159 - H * 4) /
     2)
1030  FOR K = LR TO LR + W * 4 STEP
     4
1040  HPLOT K,0 TO K,189: NEXT
1050  FOR K = UD TO UD + H * 4 STEP
     4
1060  HPLOT 0,K TO 279,K: NEXT
1070 X = LR + 2:Y = UD + H * 4 -
     2: HPLOT X,Y
1099  REM   SAVE POINTERS TO CURR
     ENT SHAPE
1100 D = I - ST: GOSUB 300:S = S +
     2: POKE S,L%: POKE S + 1,H%
1199  REM   INPUT MOVES
1200  POKE 34,0: HOME : VTAB 22: INVERSE
     : PRINT "   COMMANDS ->  LEFT  R
     IGHT  UP  DOWN      PLOT  MO
     VE  SEE   -> ": NORMAL
1210  VTAB 21: HTAB 1: PRINT "SHA
     PE #";N;" OF ";NC;" SHAPES -
      ";I - ST;" BYTES USED"
1220  VTAB 23: HTAB 27: PRINT "CO
     MMAND? ";: GET A$
1230  IF A$ = "D" THEN Y = Y + 4:
     C = 2: GOTO 1400
1240  IF A$ = "U" AND Y > 0 THEN
     Y = Y - 4:C = 0: GOTO 1400
1250  IF A$ = "L" AND X > 0 THEN
     X = X - 4:C = 3: GOTO 1400
1260  IF A$ = "R" THEN X = X + 4:
     C = 1: GOTO 1400
1270  IF A$ = "P" THEN F = 1: HCOLOR=
     7: GOSUB 100: GOTO 1210
1280  IF A$ = "M" THEN F = 0: HCOLOR=
     0: GOSUB 100: HCOLOR= 7: HPLOT
     X,Y: GOTO 1210
1290  IF A$ = "S" THEN F = 0: GOTO
     2000
1300  GOTO 1210
1399  REM   PLOT CURRENT POSITION

1400  HPLOT X,Y: ON F GOSUB 100
1410 A(J) = F * 4 + C:J = J + 1
1420  IF J < 3 GOTO 1210
1430  IF A(2) < 4 AND A(2) > 0 GOTO
     1500
1440 Q = A(2):A(2) = 0:KEEP = 1
1450  IF A(1) THEN 1500
1460 KEEP = 2
1470  IF  NOT A(0) THEN A$ = "X":
     KEEP = 0:J = 0: GOTO 2140
1499  REM   COMPILE MOVES INTO BY
     TES AND STORE
1500  POKE I,A(0) + A(1) * 8 + A(
     2) * 64:I = I + 1
1510 A(0) = 0:A(1) = 0:A(2) = 0:J
      = KEEP:KEEP = 0
1520  IF J THEN A(J - 1) = Q
1530  GOTO 1210
1999  REM   LAST 2 BYTES OF CURRE
     NT SHAPE
2000  POKE I,A(0) + A(1) * 8 + A(
     2) * 64: POKE I + 1,0:I = I +
     2
2010 A(0) = 0:A(1) = 0:A(2) = 0:J
      = 0
2099  REM   DRAW SHAPE
2100  HGR : HOME : VTAB 22: INVERSE
     : PRINT "  TYPE   'X' TO CAN
     CEL  'E' TO ENTER  ": NORMAL
     : POKE 34,22
```

```
2110  HOME : HTAB 5: INPUT "SCALE
    ? ";A$
2120 M =  FRE (0)
2130  IF A$ = "E" THEN 1000
2140  IF A$ = "X" THEN N = N - 1:
    S = S - 2:I = TI: GOTO 1000
2150  IF A$ = "S" THEN  TEXT : HOME
    : GOTO 3000
2160 SC =  VAL (A$): IF SC < 1 OR
    SC > 255 THEN  PRINT "SCALE
    MUST BE BETWEEN 1 AND 255";:
     FOR Z = 1 TO 1500: NEXT : GOTO
    2110
2170  VTAB 23: HTAB 24: CALL  - 9
    58: INPUT "ROTATION? ";R: IF
    R > 64 THEN  PRINT "MAXIMUM
    VALUE FOR ROTATION IS 64";: FOR
    Z = 1 TO 1500: NEXT : GOTO 2
    170
2180  ROT= R: SCALE= SC: HGR : HCOLOR=
    7: DRAW N AT 139,79
2190  VTAB 21: CALL  - 868: PRINT
    "SHAPE #";N;"   SCALE ";SC;"
      ROTATION ";R
2200  GOTO 2110
2999  REM   STORE TABLE LENGTH AN
    D SAVE TABLE
3000 D = I - ST: GOSUB 300
3010  POKE 0,L%: POKE 1,H%
3020  PRINT "LENGTH",D,"$";: GOSUB
    200: PRINT H$
3030  PRINT "BEGINS",ST,"$";:D =
    ST: GOSUB 200: PRINT H$
3040  PRINT "ENDS",I,"$";:D = I: GOSUB
    200: PRINT H$
3050  VTAB 8: PRINT "TO SAVE SHAP
    E TABLE ON TAPE, ENTER
    MONITOR (CALL-151) AND TYPE:
    -": PRINT
3060  PRINT "  0.1W ";
3070 D = ST: GOSUB 200: PRINT H$"
    ,";
3080 D = I: GOSUB 200: PRINT H$"W
     "
3090  PRINT : PRINT "USE SHLOAD T
    O RECOVER"
3100  VTAB 17: PRINT "TO SAVE ON
    DISK TYPE:-"
3110  VTAB 19: PRINT "  BSAVE   (
    TABLE NAME)   ,A";ST" ,L";I -
    ST
3120  VTAB 21: PRINT "TO RECOVER
    USE 'SHAPE-PROG' FROM THE
     BASUG INTRO. DISK"
3130  END
3999  REM   INITIALISE
4000  DIM A(2)
4010 UPPER =  PEEK (115) +  PEEK
    (116) * 256 - 50
4020 LOWER =  PEEK (105) +  PEEK
    (106) * 256 + 300
4099  REM   DATA INPUT
4100  SCALE= 1: ROT= 0: SPEED= 25
    5: TEXT : HOME
4110  PRINT "NORMAL CHARACTERS AR
    E BASED ON AN": PRINT "ARRAN
    GEMENT OF 5 X 7 CELLS WITHIN
     AN": PRINT "OVERALL GRID OF
     7 X 8 CELLS": PRINT : PRINT
    "YOUR GRID CAN BE FROM 1 X 1
     TO 69 X 39"
4120  VTAB 7: CALL  - 958: INPUT
    "HOW WIDE IS YOUR OVERALL GR
    ID? ";W: IF W < 1 OR W > 69 THEN
    4120
4130  VTAB 9: CALL  - 958: INPUT
    "HOW HIGH IS YOUR OVERALL GR
    ID? ";H: IF H < 1 OR H > 39 THEN
    4130
4140  VTAB 12: PRINT "THE SHAPE T
    ABLE CAN HOLD 255 CHARACTERS
    "
4150  VTAB 14: CALL  - 958: INPUT
    "HOW MANY SHAPES IN YOUR TAB
    LE? ";NC: IF NC < 1 OR NC >
    255 THEN 4150
4160  VTAB 18: PRINT "THERE IS SP
    ACE BETWEEN:": PRINT : IF LO
    WER <  = 8191 THEN  PRINT LO
    WER;" AND 8191",
4170  IF UPPER >  = 16384 THEN  PRINT
    "16384 AND ";UPPER
4180  PRINT : VTAB 22: INPUT "WHE
    RE SHOULD YOUR TABLE BEGIN?
    ";ST
4199  REM   START OF SHAPE TABLE
    & STORE ADDRESS
4200 S = ST:I = ST + NC * 2 + 2:D
     = ST: GOSUB 300
4210  POKE 232,L%: POKE 233,H%: POKE
    ST,NC: POKE ST + 1,0
4220  RETURN
63979  REM  **********************
63980  REM  *                    *
63981  REM  *  SHAPEDRAW  2.2    *
63982  REM  *                    *
63983  REM  *       SEE          *
63984  REM  *    PRACTICAL       *
63985  REM  *    COMPUTING       *
63986  REM  * SEPT. AND DEC 80   *
63987  REM  *                    *
63988  REM  *                    *
63989  REM  *     FURTHER        *
63990  REM  * MODIFICATIONS BY   *
63991  REM  *                    *
63992  REM  *  PETER L. CAVE     *
63993  REM  *  "BRIDGEMOUNT"     *
63994  REM  * 2 LITTLE HALLAM    *
63995  REM  *      HILL,         *
63996  REM  *    ILKESTON.       *
63997  REM  *  DERBY DE7 4LY     *
63998  REM  *                    *
63999  REM  **********************
```

reader's letters

East Ham

Dear David,

A point which may be of interest to Hardcore readers.
I bought my APPLE at Xmas 1980. The power pack went on the blink in Feb 81 and was repaired under guarantee.
It went again in April 82, whilst doing nothing but sitting there switched on.
The people who supplied it could only offer to take it in, send it to APPLE and either have it repaired (estimated 4-5 weeks) or to supply an "exchange unit" (3 weeks).

As I need the APPLE for my work, I tried Guestel Ltd. who were very helpful but could only offer a replacement unit on an exchange basis (which I collected from them on the same day and this got me out of trouble).

The staggering thing, however, is the price. The exchange unit is listed, according to Guestel at £175 plus VAT, provided it has not been opened, in which case it cannot be exchanged. I am still trying to find out how much it would cost me to retrieve my old one so that I can repair it for a spare.

The EXCHANGE PRICE would buy a complete Sinclair 32K colour computer plus a spare power supply. Now I know that Sinclair do not have the greatest reputation for reliability but my experience of the APPLE (I have also had trouble with the disc drives and lost the bleeper during this time) isn't that good either.

Can anybody justify the cost of the power pack, when everybody else seems to manage OK with conventional power supplies for about $40 retail?
I can assure anyone with the same problem, and with the time to do it, that a conventional power pack works the APPLE OK, can be made to fit into the space available, does not give the same radio/TV interference, and can be built for £30 or less, buying the bits at retail prices. It would be worth somebody's time to market such a unit, if my experience is anything to go by.

Yours

Ted Lepley

Brighton

Dear BASUG,

Thanks for the UPDATE. You asked for views on what BASUG should be, and why people don't attend courses. I can only speak for myself, but here goes, for what it is worth.
Seems to me that the relevant point is that most members are not hobbyists, and maybe there should be a separate group for those that are. Take myself for example. I am a professional writer and use my APPLE almost exclusively as a wordpro system. This may seem crazy in view of the availability of better wordpros for the price, but I initially bought this to be compatible with a colleague who is co-author of some of my writings and had an APPLE already.

When I bought the system I thought "Great I can play games and stuff too". In fact, after the first week, I never have.

As long as I can process words, I'm happy, and I imagine lots of other members are single users who have a similar attitude. I don't need machine code or PASCAL or whatever; but I found BASUG of immense value in putting me in direct contact with Ian Trackman, when I found "Go-Between" wouldn't operate with my printer card-- this led to Ian's report in the last HARDCORE which no doubt helped lots of other people with this problem.
So what I am really after is specific information about better word processing with the APPLE II. I get this out of the magazine, skim the rest of it and throw it out. For this, the subscription is well worth the price and I am the owner of a rugged, effective wordpro that does all I want it to do at a modest cost, and with which I can work on files sent by my colleague who is more involved with things like machine code and slips me the occasional subroutine that I use, cookbook style, without having the faintest idea why it works -- like the routine to count the number of words in an article.
I hope that explains why I -- and presumably dozens or hundreds like me -- don't to seem super-enthusiastic about all of BASUG's diverse activities. I've my own little niche and I'm happy in it. What I would like is more information on improving Apple Writer but even there I'm contented enough not to have bothered with another card so that I can run the superior Electric Pencils and the like. Maybee I ought to get more memory though!

Best Wishes,

John Gribbin

Stratford
London E15

Dear John,

In your article about enhanced mode on EPSON you asked if anybody knew why a driver on page $3 often gets lost when you return to the Applewriter Editor. My investigations show that (1) it only gets overwritten if you use the back arrow "vacuum cleaner" key; (2) if you erase > 256 characters it always gets lost; (3) if you erase only a few characters it usually doesn't get lost. Further investigation shows that Applewriter's 256-byte "deletion buffer" runs from $2DO - $3CF, thus explaining the above observations.

.... Have you seen Applewriter II? If not, GET HOLD OF IT SOON; it's really specacular. Everything that was horrible about Applewriter 1.1 has been fixed, and there are innumerable added features. I am very keen to use it, but for one disastrous problem: there is no provision for using a RAM printer driver (you can only specify a slot number for output). I am more or less committed to using a complicated driver I wrote to give underlining, super- and subscripts, and some Greek letters on EPSON MX-80 so the problem is rather acute.

Applewriter II is thoroughly protected against code-patching (it erases itself from memory on exit) and the only solution I can think of is to put my driver in I/O locations $CN00 - CNFF and $C800 over the $C8FF. The Applewriter II would call $CN00; my driver would take over the $C800 space. Do you know how this could be done? Is it possible to make up a card with just this little bit of RAM for a vacant slot?> I would pay large sums for a solution!

Looking ahead a little, it is clear that Applewriter 1.1 is dead. Presumably there will be an interest in developing a "Go-Between" for the new version, so my questions above have a wider implication that just my own difficulty.

P.S. Applewriter II is highly addictive. Be warned.

P.P.S. The shift-key mod of Type-Right doesn't work with Applewriter II although it is supposed to support shift key mods.

Yours sincerely

Robert Purves

(Ed. A quick survey of Applewriter II and Type-Right users around the office shows that the shift key mod does work for us. But we are looking into it).

Croydon

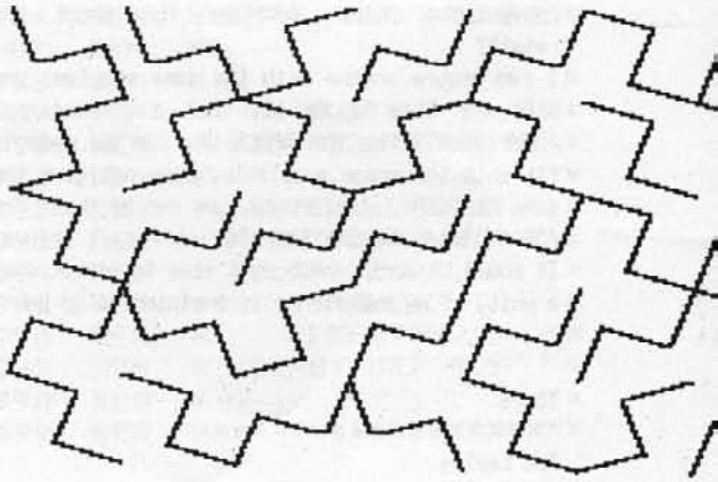
Dear Sir,

I write this note, I hope, correctly set out for immediate photocopying into your Letters page, hopefully to save you time.
It is also to make the point that most of the dot matrix reproduction employed for /sic/ is just ghastly. In this day and age we should surely do all we can to get the job done properly.
It is argued that this problem is one of cost. Rubbish. My golfball, an Adler SD 4035, cost £1100 two years ago when most others were twice the price. It was just a case of spending some time looking round properly. And don't forget to insist on the "£" symbol.
Prices have come down considerably since then, and some colleagues and I are doing another such survey: we hope to pass on our results when available.

Yours, & c.

Paul Vernon

(Ed. Many thanks for your thoughtful letter and attempt to format it for insertion. Much appreciated. Unfortunately, as you will see, we had to retype because you missed the column width by 4cm. We couldn't quite gather whether you wanted to say that the typeface used in Hardcore is ghastly. That is a new one on us. We editors have so far stuck to the Centronics/Epsom printers, not only because many members have and use them for contributions, but also because we are not entirely convinced by your claim that anything is ipso facto 'better' than a dot matrix. To be brutally frank, for instance, your expensive machine's shaded face golfball makes a very uneven impact. But let's turn it over to our readers and help Paul's survey. What do you think of our layout and presentation? Is it really time for us to turn to our daisywheel printers?)



Norwich

Dear BASUG,

Whilst reading the Apple-Writer article in a recent Hardcore it occured to me that the majority of BASUG members would probably not have RAMCARDS in their machines. I also felt that the majority of BASUG members would not always need all the buffer space available for Apple-writer.

I therefore suggest that one of the experts on the above program comes up with the patches (and explanations for users of versions already patched for Lower Case Chips!) which would chain the TEDITOR and PRINTER programs together in memory. By my calculations the programs both occupy 4.5 K. of memory which should allow plenty of room for the average piece of text with a "FAST VERSION".

In the process of putting together the patches two bugs which I find very aggravating could be fixed perhaps.

1). When I use 'C' to Continue printing from the printer menu I get the last used FILENAME or Search / Replace as a page heading. I suppose the entry of a space as the heading might cure this , but why should this be necessary ?.

2). When using multiple drives the Apple-Writer will return to the original Drive for the PRINTER program (even after saving a file called "FILE,D2) However, if the user has separate controllers the Apple-Writer will not default to the original SLOT.

To add to my suggestions for THINGS TO DO can I suggest the following be published.

A). A short program to format disks for data (allowing the early tracks to be used by patching the VTOC. to show them as unused. I appreciate that ZERO ZERO indicates something special , but would like to have the true 140K available for data disks).

B). A patch to enable the user to ignore "WRITE PROTECT" error messages. The patch would allow the re-use of old Master Disks (without resorting to scissors)).

C). A patch to stop non-present files being opened automatically. The modification of the parameter list would avoid spurious CATALOG entries and loss of disk space. It would have to be invoked before accessing files and repaired afterwards in case the file was genuinely needed if not found.

Graham Rubens.

(Ed. Yes, you've certainly come up with some good ideas, Graham. Who's going to be first with the solutions?
The CONTINUE problem shouldn't happen, I use it frequently and have never had the problem).

John Sharp

Bromley Kent

Dear BASUG,

I recently purchased an APPLE III. Could you please put me in touch with any individual or organisations that run tuition courses on an APPLE III so that I may better learn the machine.

W.G.Chapman

(Ed. We would gladly run one if we thought there were enough call for it, but in view of the trouble we have encounteredin getting support for APPLE IIE courses I do not think we could even contemplate it. Could we lend our voice to yours: is anyone doing anything for the APPLE III?)

Fareham, Hants

Dear Sir,

I have an interest in Amateur Radio. My call sign is G3YMS.

I work SSB on HF and FM on VHF.
I would like to work RTTY on HF?VHF utilising the APPLE as the terminal.
If there are other members with similar interests, would they care to contact:-

John H Ison, G3YMS
[illegible]

Nottingham

Dear Sir,

I see that in BASUG you are looking for someone to coordinate an Amateur Radio SIG. I would be quite keen to do this, although I should explain that I am rather heavily committed timewise with various other Amateur Radio affairs, but could start by commenting on some Amateur Radio Programs that I have.
My call sign is G$AFJ.

Geoff Dover

(Ed. Well Geoff and John, we know there are a lot of you Radio Hams out there, and there are many more on the sideline with a passing interest, as judged by the response to the station set up by Derek Turner, Mike Watson et al at APPLE 82. I think you are probably doing something already, but are too busy to let us know.
So yes, do get something going, but do let the others know what it is!)

Watford

Dear Basug,

First of all, let me congratulate you on running a fine club and producing an excellent magazine in "Hard Core".

I have been meaning to write to you for some time but somehow never got round to it, so here are a number of points / questions / info. / etc. which have accumulated over the past year or so of membership.

1) I have had my Apple II plus for almost two years now, but only recently discovered a piece of information about the Reset switch, which many users may find useful.
Many articles have been written about the Apple, complaining about the Reset key being too close to the Return key, thus allowing accidental resets. (I myself have found from experience that actually this only happens once in a blue moon because the "stronger" spring underneath the RESET key repels all but the deliberate presses).
However, upon reading a miscellaneous sheet of paper supplied with our latest APPPLE at work, I discovered that at the flick of a switch, the RESET function can only be obtained by pressing the CTRL & RESET keys together.

The switch in question is on the secondary keyboard circuit board (the smaller one) in the left hand corner, labelled S1.
2) When I received the Introductory Disk, I encountered a problem with "HAUNTED CAVE". The program crashes with a 'xxxBAD RETURN ERR/ STOPPED AT 3010", whenever you have 'all your treasures stolen' by one of the monsters.
I examined the program and traced it through but could see absolutely no way it could happen , but it did!!

However, I recently read an article in CALL APPLE's, 'ALL ABOUT APPLESOFT' (p 100), which describes a bug in the FP BASIC interpreter which sometimes causes 'RETURN WITHOUT GOSUB' errors in some programs. I assume that there is a similar bug in the INTEGER one as well. I haven't found out why but I have found out how to make it work.
Simply change the variable in the loop in lines 3940,3945,3950 to use a new varaible (say ZI) instead of the variable I used.
It could be that it is allocating variables over a page boundary.

3) I have also found a bug in the 'LOST DUTCHMAN'S GOLD' program on library Disk 20. When you 'EXAMINE RIFLE' it crashes with a syntax error due to a missing colon in line 1470 before the <RETURN>.

4) On Disk 18 'SPELUNKER' and 'THE MAZE' I had trouble since the programs loaded lines of garbage when they listed.

5) I have bought 8 games disks from the library, and although I haven't the time to do a complete review of each one, the following table gives some idea of my opinion.
!1j

DISK	MARKS OUT OF TEN	STAR PROGRAMS
10	7.5	Game of Genius, DUBBLE-1, Star Attack
15	5.5	Air Attack,Space War V
18	5	Oregon Trail
20	9.5	Lost Dutchman's Gold, Mini Golf, Collision
29	6	Sevens, Othello
33	7	Hi-O,Horse Race, Cribbage
40	8	Apple Wars, Space Pilot, Space War
E1	7	Eamon

All contain at least a couple of quality programs, although some contain junk as well, so making them well worth the maegre cost.

6) I also own Z8000 single board computer (made by AMD). It has an on board monitor/assembler in EPROM. However, the assembler is not very good and does in fact contain a few bugs.

I have therefore been trying to obtain a decent Z8000 cross-assembler to run on the APPLE, either under DOS 3.3 or CP/M.I have however, had no luck at all in finding one. I have tried many s/w retailers including PETE & PAM, LIFEBOAT, SPIDER S/W etc. (I would also like to obtain a Z8000 disassembler as well).

I would therefore be very grateful indeed, if you could let me have any information at all about companies or any BASUG members who might be able to help in this matter.

Yours programmingly,

M. Bryant

Surrey

Dear Sir,

I have recently joined BASUG and would just like to say how much I enjoyed the introductory disk (which I hadn't been expecting). The LIFE program itself has saved me the membership fee immediatly.
Do you have, or can you refer me to any reviews of other software, like the review in Hardcore on the statistics disk No. 32.

Although you do not publish membership lists, I would like to make contact with other nearby members, so I would be happy if you passed my name and address on.

From the aforementioned review it seems that some of the programs in the library might be capable of improvement. Would a 'significant' improvement/rewrite of a program already in the library be eligible for inclusion ? Do you accept single programs or do you want only complete disks full of programs and documentation like the introductory disk ?
Thank you again for the introductory disk, unravelling/study of the programs will keep me absorbed for weeks of evenings.

Rod Price
[illegible]
TW10 7UE

(Ed. Thanks for the comments on the introductory disks. You seem to have got on OK. Some people are having a little trouble with LIFE, and the HAUNTED CAVE. Again there is help in the early HARDCORES. The HAUNTED CAVE does need some attention, and some guide to use. Any volunteers to write a piece on how to play adventures, very very basic. The EAMON users should have someone among them who could help the beginners, they often do not know the simplest commands.

On the subject of Eamon disks, we have had problems all along the line copying them; in particular the text files. They are too long. Has anyone any idea why?)

Dear John,

I enjoyed reading your article on Epson printers, and am grateful to you for telephoning me a few weeks ago concerning my enquiry.

I have the following items for sale and wonder if you know of anyone who might be interested in buying.

1 Apple Eurocolour card ....£50 o.n.o.
1 Epson interface with graphics (DAPL ROM) ..... £70 o.n.o.

In addition I am looking for an Apple II disc drive without controller. Perhaps you know of someone who has one who is looking to sell?

I know you are kept snowed under with paperwork so don't bother to reply unless you are able to help.

Dr. W.A. Telfer Brunton

West Molesey

Dear Mr Sharp,

I have recently acquired an Apple II computer. I have had it for two months and some of the parts still puzzle me. Therefore I am writing to see if there is anyone locally who could come and see me to explain some of the theories.
I feel that I should explain that I am physically handicapped and I would appreciate any help that you could give me.
Another problem I have is that I have a number of programmes which were discussed in "Practical Computing" magazine which I would like to try out on my computer but it would take me a very long time to programme them into my computer, owing to the fact that I type with one toe.
Please let me know if there is anyone who could help me.
Yours sincerely,

Hugh Slater

(Tony Williams writes: We do have an SIG for handicapped people with disabilities as severe as yours, Hugh. As a first suggestion why not contact Glyn Vernon at the Spastic Society's Professional Workshop in Milton Keynes. When I interviewed him for PCW last year Dick Boydell was using a virtually unmodified Apple for programming but there are various simple modifications for making typing in listings easier. There are other Apple users in West Molesey and I am sure they will contact us and offer to help out.)

Ealing

Dear Sirs,

I'd like to have more information about your literature library and how can I borrow articles from it, and more important what kind of information can I access.?
I am a proud owner of an ITT 2020 and I should inform you and your members (ITT owners) that all the service manuals and disk tests can be obtained from ITT .

Sincerely Yours

C Santos

Member 82051191

(Ed. We nearly didn't print your name because we couldn't read it. It was only the number on the database that saved the day. Which only goes to show that membership numbers have some use after all.
You have touched on two problems which affect all members. A lot of information can be culled from past issues of Hard Core - such as the ITT address to write to for the service manuals, etc.

They also tell you more about what is available in the library, how to get them etc. The library list is obtained by writing to the literature librarian. You then pay post on what you borrow. We have APNOTES from the IAC in the States, and other user group magazines. There are some books, articles and all sorts of other goodies.

Well how do you newer members find all this information out? By buying every copy of HARDCORE you can get your hands on. The complete set for last year is £7.20 including post.)

1 Brook Close
Ruislip
Middx
HA4 8AF

Dear David,

COMPUTER APPRECIATION WEEK RUISLIP

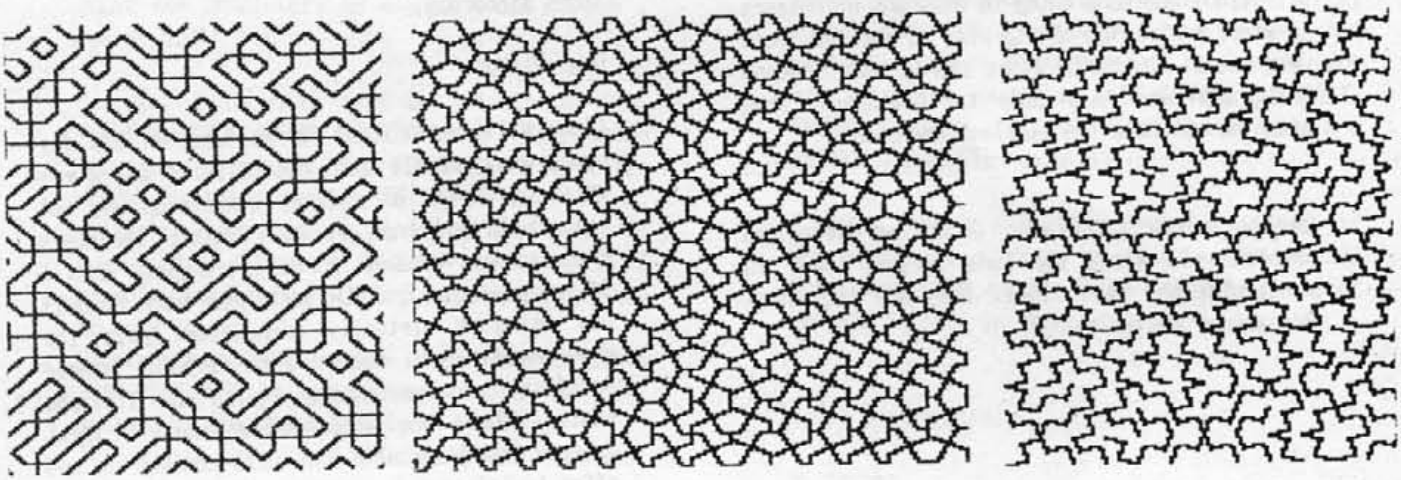
We would like to extend an invitation to BASUG to our Computer Appreciation Week from the 16th to the 21st of August 1982.
Monday and Tuesday are for Business and Professional Applications, the rest of the week will be devoted to EDucational and Personal use.

The objective of the week is to stimulate a widespread interest in Computers and computers. We welcome contributions in the three areas above. Our goal is not that people should buy computers, but that they should be sufficiently stimulated to desire to explore the full benefits of the world of computers.
Computer Town Ruislip is holding the Computer Appreciation week in association with and at the Ruislip Library.

I would be grateful if you would give this a wider coverage.

F.A.Fadipe
Project Leader

(Ed. Can any members in the Ruislip area help out? The Computertown concept is one we would like to take part in if we only had the time. The week should be a good focus for starting a local group.)



APPLESOFT SURGERY

by SHARAT MUNJAL

The purpose of this article is to illustrate how Applesoft works on program lines and to give a breakdown of the structure of a program line. The information provided here should allow you to examine your programs and correct any corruption, and also allow you to insert illegal statements into a program.

APPLESOFT works on the basis of an interpreter, as do most microcomputer implementations of BASIC. The interpreter is a program which resides permanently within the Apple. It is the interpreter's function to read in program lines and perform whatever actions are necessary, depending on the content of the program line. To the interpreter,the program is only data in a special sequence, upon which it has to operate. This means that we should also be able to get an idea of what a program looks like to an interpreter. However, normal commands from within Applesoft will not allow us to do that. So we have to use the monitor to look at the actual program statement lines.

Certain rules have to be followed, to allow an interpreter to perform correctly. These rules govern the starting location of the program,its length and more importantly, the syntax of each line within the program.

All BASIC statements are stored in a binary form within the computer, in a manner similar to all the other pieces of information that it stores. This means that each character is converted to its binary equivalent when you type it in and converted back when you display it. In a similar way, all Applesoft commands are stored as binary numbers. To save space, each command word has been allocated a unique number called its "token" . Each token is a one byte long. The space saving should be immediately obvious. Instead of using 6 bytes for the command NORMAL, we only use one. A list of all Applesoft tokens is on Page 121 of the Applesoft Manual.

An Applesoft program line appears as shown below:-

The 6502 processor operates on all addresses in reverse order in the two bytes that are available for each address. This means that an address such as $FDED will be stored as $ED,$FD. Applesoft program lines start building up from LOMEM (Default is $800). Applesoft also requires that the first byte of any program must be a 00. Therefore the actual program line usually starts from $801 (2049 Decimal).

Program lines within Applesoft are linked to each other by a method of 'chaining'. This means that each line within an Applesoft program carries the starting address of the next line, as shown in the above diagram. This chaining is essential, since this is the only way that Applesoft knows where each particular line is located. Every time it comes across a GOTO or GOSUB instruction it starts from the beginning of the program (usually $0801) and follows the links until it finds the line that it is looking for. (This is the reason why you should have all your frequently accessed routines at the beginning of the program. Each time the interpreter comes across a GOSUB it starts from the beginning. If you have your routines near the front, it doesn't have to travel a long way down the program looking for the line number).

Each program line begins with a two byte address field which contains the starting address of the next line. This address is not related to the line number of the next line. It refers to the position within memory. The next two bytes contain the current line number. With two bytes the highest number you can store is 65535. However, it only allows you line numbers up to 63999.

The actual program starts from byte 5 of the line. This means that the first character of any program usually appears at 080. All command words are shown in their tokenised format. All strings within the program which are between quotes are maintained as they are and the end of a program line is indicated by a $00. At this point the interpreter stops executing and gets the 'next' line of the program.

In addition to tokenising a program, an interpreter also takes care of the placing of data and variables for the program and maintains the pointers for these. Data strings created during the running of the program start building down from HIMEM, while the actual program lines build up from LOMEM. A reasonably simple table is shown on page 127 of the Applesoft Manual.

Applesoft maintains within page zero of the memory certain pointers which are connected with the program currently in memory. The relevant ones are shown below:-

LOCATION	USE
$67-68	Beginning of program.Usually set to $0801 for ROM based machines
$69-6A	Start of simple variable space. It is also quite frequently the end of program 1 or 2
$6B-6C	Pointer to beginning of array space.
$6F-70	Start of string storage. String storage starts from here and goes on till HIMEM
$73-74	Highest location in memory available to Applesoft plus one.
$AF-B0	End of program.

Keep in mind that all these addresses are held in reverse order,which means that when you read them, read the second byte first, e.g. $67-68 may hold $0801 for the start of the program. You will find that $67=01 and $68=08.

You must access and change this data through the monitor. The instructions for the use of the monitor are supplied either in the Red book or the Apple II Reference Manual.

I hope to produce an article such as this for Integer Basic in the near future.

HOW TO SUPERCHARGE YOUR APPLE

By Richard Teed

There are two things needed to do this. The first is a BASIC compiler such as the EXPEDITER. The second is an arithmetic processing unit such as the AM9511 from California Computer Systems. Although the second is expensive it can make dramatic changes to execution times, for example:

```
100 FOR N=1 TO 5000
110 A=A+SQR(N)
120 NEXT N
130 PRINT A
```

This program takes 4:24 minutes in normal Applesoft. Compiling it reduces the time by 12 seconds to 4:12, this is not much of a saving because the compiler uses the floating point routines in the interpreter - the time saved comes from the compiled code knowing where to find its variables and commands more quickly than the interpreter, the larger the program the greater the advantage of compiling.

Running the above program on the floating point processor reduced the execution time to just 24 seconds. Now the compilers saving becomes significant, using both drops execution time to slightly over 11 seconds.

To give you a comparison a Data General Nova 4 takes about 2 seconds to run the same program in FORTRAN.

There are a few problems that hav occurred with the arithmetic processsor and they are: Applesoft is RAM so you lose 10K but you can get round this by updating a copy of Applesoft on a RAM card. The second is more serious and is to do with hi res graphics - the program just seems to think its hit an end command - there's no telling where, it just stops (the same place for each program). Thanks to the Expediter a trace should show the problem.

A final word on the Expediter, I have heard of a program that did a string sort and took about an hour in Applesoft; compiled it took slightly under a minute.

APPEND

The Final Solution - Don't Do it

by Hedley G Wright
Argyll Research and Analysis

The Applesoft Disk II DOS command APPEND has several faults which make it useless for serious purposes. The object of this article is to demonstrate the use of sequential files in Applesoft by simple routines that provide the facilities of APPEND and improved speed of working.

Unlike binary files and Applesoft program files, textfiles contain no information about their exact length. The reason why they were designed this way is possibly related to Apple DOS's economical use of disk space in random access textfiles. The end of a textfile, or a textfile record in the case of random access files, is marked by the writing of an ASCII zero, the End Of File (EOF) Marker, provided that no records of the file have previously been written on the disk sector in use. Thus if a file is overwritten on itself in a shorter form, without deleting the original, then no EOF will be written. Sometimes however the EOF marker is omitted or is not overwritten by records subsequently added to the file by the APPEND command.

APPEND is intended to find the end of a sequential textfile rapidly so that more data can be added, and this it conspicuously fails to do. It functions by starting at the beginning of a file and reading serially through until it finds the EOF zero marker. Reading then stops and the next WRITE to the file should cover up the old EOF marker and write a new one when the file is CLOSEd. APPEND, however, will fail to function correctly if the EOF was never written or if an old EOF marker was not covered up in a previous APPEND operation. Various "fixes" have been proposed (read Neil McFerran in HARDCORE Vol.1 No.4) but APPEND is still a menace to its users and a time-waster. In a search for reliable methods of appending information to the end of sequential files various alternatives were tried. The most obvious and simple one is to keep a count of the number of fields in the file at the start in a fixed length record zero, which can be rewritten as the file grows; then it is only a simple matter of skimming through the file with the POSITION command, counting the field-end RETURNS, to reach the end of the file and start writing. This technique has the same damning flaw that makes APPEND useless in practice. It takes too long.

If DOS does not keep a record of the length of a text file then the obvious answer is to fill this want by writing the length into record zero. The little-used though immensely valuable BYTE parameter, ",B", assists in handling this. If all the data in a file are written as text strings, the length of the file can be recorded by keeping a tally in the form "BYTE = BYTE + LEN(entry$)+1"; the "+1" is required to account for the terminal RETURN. After new data have been written the file is CLOSED, reOPENed and the byte tally is written as a fixed length string in record zero i.e. "PRINT RIGHT$("0000" + STR$(BYTE),5)". When the file is next OPENed to enter more data, record zero can be read and a WRITE can take place at the end of the file immediately. Move-by-move details of this simple implementation will not be given as a more sophisticated approach is recommended.

The shortcomings of the simple byte count scheme are that all data have to be written as text strings since it is not possible to determine directly the length of real variables, and more importantly, the BYTE parameter is restricted in value to less than 32768 ($8000), which is unexpected and not documented.

The program printed here demonstrates an effective way of appending; the routines for OPEN, WRITE, READ and CLOSE are illustrated and it is not necessary to read any further to be able to extract and use them in other programs. It is strongly recommended that they be adopted if long files are being used; there will be no difficulties in appending rapidly to files up to the full capacity of a DISK II floppy.

The essence of the program technique is to monitor the file length by inspecting the File Manager Work Area in DOS, note carefully that this is not the same as the File Manager Work Area Buffer for the file. Possibly some APPEND errors originate in faulty exchange of information between the Work Area and the Work Area Buffer. To follow this article in detail it is worth consulting "Beneath Apple DOS" by Worth and Lechner, page 6-15.

In a 48K system the Work Area starts at $B5D1 (46545 decimal). This address is calculated from page 3 parameters for any size of system in line 100 of the program and, as written, will work for any size of system except those with DOS relocated in a memory expansion or language board; in such cases DOS cannot be accessed directly from BASIC, machine code routines would be required. The basis of the calculation is that the work area starts $16 above the File Manager Parameter List: the address of the start of the Parameter List is given by the contents of $03DD/03DE (high byte) and 03E0/03E1 (low byte).

The File Manager Work Area maintains information about the file currently being processed; the following positions above its start contain the most pertinent information.

Start plus:-

$13/14 the current file sector being accessed

$15 the byte position within the sector

$17/18 the ",L" parameter of a random access file

$19/1A the R A file record being accessed

$1B/1C the byte position within the record.

The actual position indicated by the byte pointers is one ahead of the last byte processed; in other words, to the potential position of the next data to be written or the EOF marker if the file is closed.

If a sequential file is being processed the ",L" is defaulted to one; consequently pointer 19/1A counts through the file in records of one byte and 1B/1C, the byte offset into the record, remains zero. Since the record pointer is only two bytes long it will return to zero after 256 (decimal) sectors of a file have been processed. 19/1A and 1B/1C thus cannot be used directly to assist in the use of the ",B" parameter in a sequential file. However, if the sequential file is opened with an ",L" parameter of 256 then $19/1A and $1B(/1C) move in parallel with $13/14 and $15 and one can count into the file using sectors/records and bytes to well beyond the capacity of a mini-floppy. There is no feature of Apple's DOS that prevents a random access file being used sequentially if the records are contiguous.

The program keeps the byte parameter up to date when the file is being written to and, although the file length is stored as a byte number, together with the number of records in a combined fixed length field in record zero, this number is recalculated and used in a ",R", ",B" format. It is not necessary to specify ",R" and ",B" for any WRITE to the file other than the first one of a batch, however, not to do so would only require extra coding.

```
10  REM DEMONSTRATION SEQUENTIAL
11  REM       FILE HANDLING
12  REM **************************
13  REM       HEDLEY G WRIGHT
14  REM ARGYLL RESEARCH
15  REM                  AND ANALYSIS
16  REM       19TH MARCH 1982
17  REM **************************

100 WRKAREA =  PEEK ( PEEK (992) +
     PEEK (993) * 256) + ( PEEK
    ( PEEK (989) +  PEEK (990) *
    256)) * 256 + 22
110 TEXT : HOME : IF WRKAREA <  >
    46545 THEN  INVERSE : PRINT
    "THIS IS NOT A 48K SYSTEM":
    NORMAL: PRINT
120 PRINT "WHAT IS THE NAME OF T
    HE SEQUENTIAL FILE": PRINT :
     INPUT FILNAME$
130 GOSUB 1000: REM * OPEN THE F
    ILE
140 ONERR  GOTO 1510
150 VTAB 10: CALL  - 958: PRINT
    "ENTER NEW DATA, <RETURN> TO
     QUIT"
160 PRINT : INPUT "DATA - ";ENTR
    Y$
170 IF  LEN (ENTRY$) = 0 THEN 20
    0
180 GOSUB 2000: REM * WRITE NEW
    DATA
190 GOTO 150
200 GOSUB 3000: REM * CLOSE THE
    FILE
210 HOME : PRINT "THE "FILNAME$"
     FILE": PRINT "IS NOW CLOSED
    ": PRINT : PRINT "DO YOU WIS
    H TO READ THROUGH THE FILE":
     PRINT "<Y> OR <N> ";: INPUT
    ANSWER$
220 IF  LEFT$ (ANSWER$,1) <  > "
    Y" THEN 310
230 GOSUB 1000: REM * RE-OPEN TH
    E FILE
240 IF FIELDNUM = 0 THEN  PRINT
    "THERE ARE NO RECORDS": GOTO
    300
250 PRINT  CHR$ (4)"READ"FILNAME
    $
260 FOR COUNTER = 1 TO FIELDNUM
270 INPUT ENTRY$
280 PRINT ENTRY$
290 NEXT COUNTER
300 PRINT  CHR$ (4)"CLOSE"FILNAM
    E$
```

```
310  PRINT : PRINT "THAT IS THE E
     ND OF THE PROGRAM YOU MAY": PRINT
     "APPEND MORE DATA TO THE FIL
     E BY": PRINT "RE-RUNNING IT"
     : END
1000  REM * INITIALISE NEW FILE O
     R OBTAIN LENGTH OF EXISTING
     ONE
1010  ONERR  GOTO 1500
1020  PRINT  CHR$ (4)"OPEN"FILNAM
     E$",L"256
1030  PRINT  CHR$ (4)"READ"FILNEM
     E$
1040  INPUT ZERECRD$: PRINT  CHR$
     (4)
1050 FIELDNUM =  VAL ( LEFT$ (ZER
     ECRD$,6))
1060 LASTBYTE =  VAL ( RIGHT$ (ZE
    RECRD$,6))
1070  IF ERFLAG THEN ERFLAG = 0: GOTO
     140
1080  RETURN
1321 MARCH1982
1500  IF  PEEK (222) = 5 THEN 153
     0
1510  PRINT  CHR$ (4)"CLOSE"
1520  PRINT "ERROR " PEEK (222)"
     IN LINE " PEEK (218) +  PEEK
     (219) * 256: END
1530 FIELDNUM = 0
1540 LASTBYTE = 13
1550 ERFLAG = 1: GOTO 3000
2000  REM * WRITE A RECORD
2010  PRINT  CHR$ (4)"WRITE"FILNA
     ME$",R" INT (LASTBYTE / 256)
     ",B"LASTBYTE -  INT (LASTBYT
     E / 256) * 256
2020  PRINT ENTRY$
2030 FIELDNUM = FIELDNUM + 1
2040 LASTBYTE =  PEEK (WRKAREA +
     21) + ( PEEK (WRKAREA + 19) +
      PEEK (WRKAREA + 20) * 256) *
     256
2050  PRINT  CHR$ (4)
2060  RETURN
3000  PRINT  CHR$ (4)"CLOSE"FILNA
     ME$
3010  PRINT  CHR$ (4)"OPEN"FILNAM
     E$
3020  PRINT  CHR$ (4)"WRITE"FILNA
     ME$
3030 ZERECRD$ =  RIGHT$ ("00000" +
      STR$ (FIELDNUM),6) +  RIGHT$
     ("00000" +  STR$ (LASTBYTE),
     6)
3040  PRINT ZERECRD$
3050  PRINT  CHR$ (4)"CLOSE"FILNA
     ME$
3060  IF ERFLAG THEN 1000
3070  RETURN
]
```

UPDATING THE DOS 3.2 MANUAL TO 3.3

By David Bolton

A large number of people appear to have upgraded to DOS 3.3 by either buying a language card, or by obtaining the P5A/P6A chips. In either case, they will not have the DOS 3.3 Reference manual, and the following notes therefore detail how this varies from the 3.2 Reference manual.

Page 38 The COPY program will now provide single-drive copying.

Page 44 The UPDATE program is renamed MASTER CREATE. It will NOT update a 3.1 or 3.2 disc to 3.3.

Page 128 Track /Sector List- Amend, inter-alia, to read:-

3/4 Not Used
5/6 Sector Base Number (counts groups of 122 sectors)
7/b Not Used

Page 129 Third Para- Delete last sentence (If a complete.....).
Fourth Para- DOS 3.3 would use 12 sectors for Track/Sector list and one for record totals 13.

Page 130 Second Para- Directory begins at Track $11 Sector $F (not $C) and allows max. 105 files.
Directory entry for one file- Relative bytes 21/22 contain sector count.

Page 132 Change Table- Byte 2 Value F
3 4
31 FF
35 F

Page 133 Track Bit Map:-

Byte	Bit	Sector	Byte	Bit	Sector	
1st	7	F	2nd	7	7	
	6	E		6	6	
	5	D		5	5	3rd and 4th-
	4	C		4	4	all spare.
	3	B		3	3	
	2	A		2	2	
	1	9		1	1	
	0	8		0	0	

Page 134 Typical Track Bit Map- Amend in accordance with above.

Page 135 There are 496 sectors available to user.
Amend all references to sector $C to read $F.

Page 166 Initialising a diskette- change reference to UPDATE to read MASTER CREATE.

Page 171 Copying a text file- refer to FID program (Appendix J).

*star trek

The full instructions for the version on BASUG Disc 25.

APPLE][TREK is a sophisticated space war game in which the player, as Captain of the Starship ENTERPRISE is sent on a search and destroy mission against the KLINGON Empire Fleet. The APPLE][computer creates the game environment, operates the KLINGON Cruisers in combat and transforms the APPLE][keyboard and display into a spaceship command console.

THE GALAXY

APPLE][TREK is played in a galaxy which is represented by a grid of 64 quadrants charted as an 8 by 8 array. Each quadrant contains 64 sectors, again in an 8 by 8 array. The sector is the elemental location in the universe and may be occupied by only one of a star, a KLINGON, or the ENTERPRISE. The galaxy is a closed space in which the opposite edges are actually adjacent to each other. Moving to Galactic East of the Eastmost quadrant, the ENTERPRISE will enter the Westmost quadrant. At the start of a mission, the APPLE places all stars, Klingons, bases, and your ship at random.

THE STARSHIP ENTERPRISE

The ENTERPRISE is a powerful craft with somewhat more firepower and energy capacity than the battle cruisers of the Klingon fleet. However, the ENTERPRISE is usually heavily outnumbered and is easily destroyed unless good maneuvering and firing strategies are used.

The ENTERPRISE has two forms of weaponry; Photon Torpedoes (PH TORPS) and Phasers. The ENTERPRISE normally carries 10 PH TORPS which may be restocked through energy conversion or by visiting a base.

The ENTERPRISE carries energy in three forms; available energy, shield energy, and PH TORPS. The available energy is used for Phasers, propulsion, and to operate the ship's sensory and display subsystems.

Shield energy is carried in ship's shields to absorb hits from attacking Klingons. If the ENTERPRISE receives an enemy hit that reduces its shield energy too low, then the ENTERPRISE may sustain damage to its subsystems (see damage report).

PH TORPS are considered as energy in determining the maximum allowable energy aboard the ENTERPRISE. Also, PH TORPS can be converted directly into energy or formed from energy. The transformation consumes energy.

The ENTERPRISE receives new energy from contact with restocking bases and from on-board Dilithium Crystals. In addition, energy reallocation between available energy, shield energy, and PH TORPS may be directed by the captain. Rendezvous with a restocking base brings the ENTERPRISE back to maximum and repairs all sub-systems. The restocking base is depleted by this action and can not be used again. The on-board Dilithium Crystals generate energy at a rate of 50 units per 0.1 stardate.

KLINGON BATTLE CRUISERS

The Klingon Battle Cruiser is somewhat less powerful than the ENTERPRISE but usually runs in squadrons of several ships at a time and therefore may have greater joint firepower than the ENTERPRISE in a battle. Each Klingon has 800 units of energy and 3 Photon Torpedoes (PH TORPS) at the beginning of a battle. The Klingon energy may be used for Phasor fire, movement, or to absorb hits from the ENTERPRISE. When the Klingon energy is reduced to zero, the Klingon is destroyed.

The Klingons have both Phasors and PH TORPS which operate the same as the ENTERPRISE (see below) except that the Klingons cannot lock PH TORPS, or convert between PH TORPS and ENERGY.

Klingon's can move one sector distance per turn during a battle. They may retreat into adjacent quadrants and become restocked at that time.

CONSOLE DISPLAY

The console display for the APPLE TREK mission is presented in four display segments. The top third of the screen displays the Galactic Record, the lower left of the screen contains a detailed display of the current quadrant, the lower right of the screen is a status display, and the center of the screen is reserved for command I/O.

THE GALACTIC RECORD

The Galactic Record is displayed at the beginning of the game or whenever any navigation command is entered. The Gal-

actic Record is an 8 by 8 array of numbers representing a summary of the number of Klingons, Bases, and stars in each quadrant of the galaxy that have been observed during the game. Figure 1 shows a typical Galactic Record. The number shown in a square of the Galactic Record should be interpreted digit by digit. The ones digit is the number of stars in that quadrant, the tens digit is the number of bases, and the hundreds digit is the number of Klingons. Some examples would be:

```
305   3 Klingons, no bases, 5 stars
13    1 base, 3 stars
4     4 stars
```

The Galactic Record contains data for each quadrant that the ENTERPRISE has occupied. In addition, the data for the 8 surrounding quadrants are presented if the Long Range Sensor is operational.

The Galactic Record display area is also used for several other utility displays such as the Probe and Damage Report.

THE QUADRANT DISPLAY

The Quadrant Display presents a detailed picture of the quadrant currently occupied by the ENTERPRISE. Figure 2 illustrates a typical Quadrant Display. The display is in inverse video. The locataion of the ENTERPRISE is indicated an "E", Klingons are shown as "K", stars as "*", and bases as "B".

THE STATUS DISPLAY

The Status Display presents a brief summary of the current status of the ENTERPRISE. The display includes current sector location, years remaining in the mission, current stardate, conditior code (green = no problems, yellow = low on energy, and red = Klingons present), shields (percentage of total energy that will go to shields), shield energy, available energy, number of PH TORPS, number of Klingons, and number of bases. The last line of the Status Display indicates the course coordinates set by the on-board computer to permit fire and move sequences.

The Status Display area is also used to display the list of possible commands whenever a non-legal command is entered.

COMMANDS

The ENTERPRISE is controlled by entering the following commands from the keyboard:

```
1 - Navigation
2 - Set Shield Energy
3 - Damage Report
4 - Phasers
5 - Ph Torps
6 - Load Ph Torps
7 - Computer
8 - Probe
9 - Self Destruct
```

The request for a command appears in the left center display area. The commands operate as described below.

1-NAVIGATION

The Navigation command is used to move the ENTERPRISE to different sectors within a quadrant, or to different quadrants within the galaxy. After pressing "1", the Galactic Record is displayed. If Warp Drive is not damaged, the question WARP OR ION (W OR I)? will appear. Warp Drive is used for movement between quadrants. A warp factor will be requested. Movement will occur to a quadrant that is that many units distant. Similarly, with Ion Drive, duration will be requested and will specify the number of sectors to move within the quadrant. With both warp and ion drive, a course is also requested. This is simply the angle or direction that you wish the ENTERPRISE to travel in. Galactic North

FIGURE 1
SAMPLE GALACTIC RECORD

```
:   :   :   :   :   :   :   :   :
:   :   :3  :104:4  :   :   :   :
:   :   :12 :8   :21 :   :   :   :
:   :   :3  :312:104:4  :   :   :
:   :   :11 :2   :3   :16 :   :   :
:   :   :1  :202:4  :5   :   :   :
:   :   :   :   :   :   :   :   :
:   :   :   :   :   :   :   :   :
```

Note: The quadrant that the ENTERPRISE is in will appear in inverse video.

FIGURE 2
SAMPLE QUADRANT DISPLAY

```
                1
                2
    *       *   3
  K             4
K           E   5
      *         6
    *         * 7
      B         8
  1 2 3 4 5 6 7 8
```

Note: Display is in inverse video.

(up on your screen) is 0 degrees, East (to the right) is 90 degrees, etc. Any number between 0 and 359 may be entered.

2 - SET SHIELD ENERGY

The ENTERPRISE maintains a certain portion of its available energy in the shields to absorb blows from Klingon fire. If the shield energy goes below 10 units, damage may be sustained to the operating systems of the ENTERPRISE. The Set Shield parameter sets the percentage of total energy that goes to shields. This value is initially 50%.

3 - DAMAGE REPORT

Command 3 will cause the top portion of the screen to display the current status of all of the ENTERPRISE systems such as Phasers, PH TORPS, Computer, etc. If the subsystem is operational, the display will indicate OK. If damage has occurred, the display will indicate the estimated time required to repair the subsystem.

4 - PHASERS

Firing the Phasers will cause a blast of energy to be shot at all targets within the quadrant. The energy is equally divided among the targets and is diminished by the distance between the ENTERPRISE and the target. The available energy of the ENTERPRISE is decreased by the amount of Phaser fire. Hits on the Klingons will decrease their energy by a like amount. Phasers can be locked on to one or more targets using the computer (see below). In that case, the energy is spread evenly between the selected targets. Phaser fire is not blocked by stars or Klingons.

5 - PH TORPS

Photon Torpedoes may be fired under manual or automatic control. Under manual control, only a single TORP may be fired. A trajectory must be input (angle is similar to course discussed above). Under automatic control, you can use the computer to lock-on to any number of targets (see computer section). The computer then directs the PH TORPS to their targets. The only disadvantage to automatic fire is that the Klingons then get to shoot first.

Photon Torpedoes will cause 500 energy units of damage if a hit is made. The torpedo will hit the first object that it encounters on the given trajectory, be that a Klingon, a star, or a base.

6 - LOAD PH TORPS

Photon Torpedoes can be converted to or from energy using this command. You will be asked how many TORPS to load. A positive number response will convert energy to PH TORPS at a rate of 500 units of energy each. A negative input will convert PH TORPS to energy at the same rate.

7 - COMPUTER

The captain of the ENTERPRISE has an APPLE-81 computer at his disposal, giving him a major advantage over his Klingon opponents. The computer has 7 command options which will be displayed whenever a non-legal input is made (such as 0 or 8). These options allow you to compute a course angle to any given set of sector or quadrant coordinates, compute a trajectory to a Klingon, lock Phasers or PH TORPS to any number of targets, lock in a course, or display ship's status. Control can be returned to command mode by entering the return command (option 7 while in Computer mode).

8 - PROBE

The Probe command is used to assess the strength of Klingons that are in the same quadrant as the ENTERPRISE. The command will display the coordinates, energy levels, number of PH TORPS, and "lock" status of all Klingons in the current quadrant. The "lock" status specifies whether or not the Klingon's Phasers or PH TORPS are locked-on to the ENTERPRISE or not. The Probe information is displayed in the Galactic Record display area.

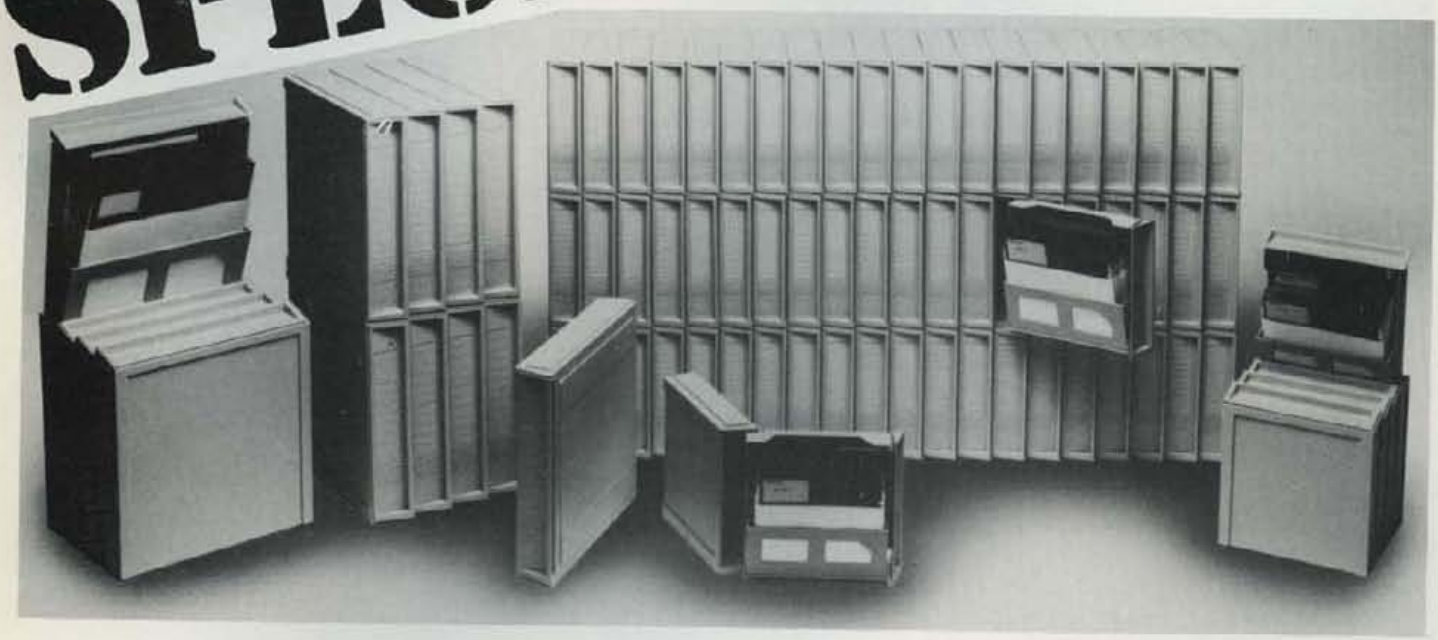
9 - SELF DESTRUCT

The Self Destruct command should be reserved until there is no hope of the survival of the ENTERPRISE. It will cause the ENTERPRISE to explode, hopefully taking any nearby Klingons with it.

END OF A MISSION

A mission will end when one of three events occurs. If all Klingons are destroyed, the ENTERPRISE has successfully completed its mission. If the time allotted for the mission runs out, the mission ends with the Klingons still threatening the empire. Finally, if the ENTERPRISE is destroyed, the mission is considered a failure. Your performance will be rated under each of these circumstances and an appropriate dispatch from Star Fleet Command will be sent. Your rating will depend on the overall success of your mission, on the number of bases used up, the amount of time required, and the amount of energy and PH TORPS used.

SPECIAL



The most versatile diskette filing system available.

This unique, modular system gives you safe, dust-free storage for your data while, at the same time, making it extremely easy to get to.

Sturdy, interlocking cases, each holding ten diskettes in a slide-out drawer, can be arranged in rows or stacks of almost any size to meet your space requirements.

As your diskette library grows, so does your storage/filing system. Any number of the modular cases can be interlocked to form a unified and convenient filing system. And, more cases can be added at any time—either horizontally or vertically.

And goes with you

When you travel, individual cases make perfect diskette carriers, and are easily removed without disturbing the rest of the system. Additionally, cases can be removed for storage in a fireproof safe, or easily rearranged to meet new space requirements.

Each case comes with two pressure sensitive identification labels. Label surfaces can be typed on or marked on with pen or pencil to show, at a glance, which diskettes the case contains. So, you can quickly find the diskette you need — when you need it

The ultimate in diskette storage, protection and retrieval

£2·75 INCL VAT

100 SUPERB LIBRARY BOXES AVAILABLE SUPPLIED ON REQUEST FREE WITH EACH BOX OF 10 AKKUTRAK DISKS SOLD (instead of normal library boxes)

BRITISH APPLE SYSTEMS USER GROUP

P.O. Box 174, Watford WD2 6NF

OFFER

Pete & Pam Computers

The APPLE Computer Specialists

Hardware & software distribution is our business — WORLDWIDE

Payment in sterling or dollars
other currencies by arrangement

Over 600 items for APPLE

From business to scientific, from education to pleasure. It's here NOW, make sure you get YOUR Copy—write or telex either of our offices now. If you're interested in Apple computers, you can't afford to be without it.

SATURN SYSTEMS

128K and 32K doards and VC— Expand The 32K BOARD
Comes with utilities to allow the movement of DOS and the use of Integer together with the ability to store subroutines on the board to be called from a main program. The final utility allows the board or multiple of boards to be used as a fast disk drive
£149.00

128K BOARD
Can be used as above with the additional facility to use the card as a fast disk drive in C/PM and PASCAL in addition to BASIC
£359.00

VC EXPAND
Is a utility that can be used with either the above two boards to give additional memory for VISICALC models, up to 146K with the 128 board—and more with additional boards
£55.00

COMING SOON
A version of VC EXPAND to allow use of VISICALC with the VIDEX 80 column board
(VIDEX 80 column board—£185.00)
(VISICALC—£105.00)

MICROSOFT PRODUCTS

MICROSOFT have written most of the BASICS for the World's Micros. As MICROSOFT'S biggest UK distributor we carry a wide range of MICROSOFT products for APPLE

TASC the Applesoft computer
True machine code programs for your APPLESOFT BASIC
£109.00

Z—80 SOFTCARD
THE C/PM System for APPLE. Over 35,000 sold to Apple users world-wide, making APPLE the most popular C/PM system
£189.00

A.L.D.S.—Assembly Language Development
System can handle 6502, Z-80 or 8080
£79.00

FORTRAN 80 £109.00
COBOL 80 £359.00

THE ENHANCER II

The dawn of a new era for the APPLE II
Introducing the ENHANCER II—a new standard which is improving the relationship between Humans and Apples. The Enhancer II can help your Apple II's keyboard become more sociable by remembering words or phrases which can be entered into the Apple by the mere touch of a key. Life can become even easier because the Enhancer II can remember what you typed while your Apple was busy talking to your disk (or doing other things). Naturally, it knows the difference between upper and lower case letters and what shift keys are supposed to do. It even knows to auto repeat any key held down. The Enhancer II replaces the encoder board making installation simple.
£99.00

THE PRICE LIST

D BASE II—from Ashton Tate
For Apple II with Z-80 softcard
A true relational database able to work on multiple files—gives you the power to use your Apple for jobs that were previously reserved for main frames.
£395.00

MICROSOFT Z-80 SOFTCARD £189.00

WORD PROCESSING

The Wordstar Family (requires Z-80)

WORDSTAR £145.00
MAILMERGE £69.00
SUPERSOFT £85.00
DATASTAR £140.00
SPELLSTAR £89.00
WORDSTAR Training Manual £19.00

MACHINE COVERS—only the best material used

Apple only	£5.95
Single Disk	£2.95
2 stacked disks	£4.45
Apple, 2 disks and 9" monitor or Apple and 12" monitor	£8.95
Apple and 2 disk	£7.95
Epson MX 70/80	£5.45
Paper Tiger 445—460	£5.45

apple computer

NEW ADDITIONS

KEYPLUS from Aids Data
A Visicalc compatible keypad with left and right arrows, space bar and escape key in addition to numbers
£99.00

CLIP-ON APPLE FAN
With forward facing Apple power switch neon lit
£49.00

COOL-STACK
A colour compatible shelf arrangement with integral fan to sit over Apple to accommodate disk drives, act as a stand for a monitor plus allow easy access to the inside of your Apple
£69.00

NEW VISISCHEDULE
from the publishers of Visicalc
A powerful project planner. Allocates all costs, specifies earliest and latest start dates prerequisites and deadlines for each task, automatic calculation and display of critical path. Can link with other Visi programs
£189.00

MATHSMAGIC
A unique number processing system designed for linear sequential calculations using a threaded interpretive language. Make your Apple a very powerful programmable calculator
£59.00

Authorised Apple Sales and Service

LONDON RETAIL
98 Moyser Road, London SW16 6SH
Telephone 01-677 2052/7341

MAIL ORDER AND DISTRIBUTION
Waingate Lodge, Waingate Close, Rossendale, Lanc. BB4 7SQ
Telephone (0706) 227011

Prices do not include VAT please add 15% to your remmittance
Postage and packing FREE

Telex No. 635740
Orders welcome by phone or telex
PETPAM G

BARCLAYCARD VISA ACCESS